UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE ESTADÍSTICAS

MÁSTER EN MINERÍA DE DATOS E INTELIGENCIA DEL NEGOCIO

TRABAJO FIN DE MÁSTER

ANÁLISIS DEL SECTOR CINEMATOGRÁFICO:

Estimación de ingresos de películas comerciales en el fin de semana de estreno en USA, mediante algoritmos lineales y de machine learning.

José Jaime Díaz García

Cotutores:

Dr. Javier Portela García-Miguel

Dr. Antonio Pareja Lora

2015

AGRADECIMIENTOS.

Me gustaría aprovechar este espacio para mostrar mi más profundo agradecimiento a Fortunato Taronna Pumilia, sin cuya valiosa ayuda todo el proceso habría sido mucho más trabajoso. Sin lugar a dudas este trabajo también es tuyo.

Contenido

INDICE DE IMÁGENES

INDICE DE TABLAS

1 INTRODUCCIÓN

1.1 PLANTEAMIENTO DEL PROBLEMA

Este proyecto de investigación nace basado en la ciencia multidisciplinar de la Minería de Datos y la Inteligencia de Negocios. Cada día se generan ingentes cantidades de información, y los medios técnicos a nuestro alcance nos están permitiendo que la capacidad de recolección de datos presente un incremento exponencial en el tiempo. Es por ello que esta relativamente joven rama profesional está cobrando más y más importancia; puesto que permitirá analizar este tsunami de información que se genera cada día.

En buena medida las redes sociales, los blogs y webs de influencia no son ajenas a esta tendencia. El comportamiento de los individuos en las redes sociales es un reflejo de su comportamiento real. Incluso, en casos extremos, el comportamiento real puede llegar a ser el reflejo del comportamiento en las redes sociales. Por lo tanto, las opiniones que se vierten en las redes sociales suelen ser un resumen bastante fiel a las corrientes de opinión de la sociedad.

El sector cinematográfico, aunque no afectado directamente como otros sectores, sí debe aprender a convivir con este nuevo sistema de información, en el que Internet en general y las redes sociales en particular, puede llegar a influir de forma determinante en el éxito de una compañía.

1.2 ALCANCE DEL PROYECTO

El punto de partida del proyecto es una base de datos que contiene información sobre el sector cinematográfico, creada a partir de la automatización en la extracción de información de redes sociales (Twitter y YouTube)[1] y webs especializadas (IMDB y BoxOffice Mojo)[2].

Partiendo de esa base de datos, uniéndola con SAS, se realizará un análisis que mejore la capacidad de predicción de la información, mediante depuración de valores ausentes o desconocidos (*missings*), atípicos (*Outliers*) o frecuencias demasiado altas/bajas entre otros posibles defectos de los datos de partida. A partir de aquí, se realizará un desarrollo de modelos, para estimar de la mejor manera posible los ingresos que una película va a obtener en su primer fin de semana al menos 10 días antes de su estreno. Para ello vamos a comenzar analizando los resultados de una regresión lineal. Posteriormente, partiendo de estas variables principales, se intentará crear y aplicar modelos de redes neuronales y árboles, y seguir así mejorando las estimaciones.

[1] https://twitter.com y https://www.youtube.com/
[2] http://www.imdb.com/ y http://www.boxofficemojo.com/

Para realizar el trabajo, se va a utilizar SAS (Javier Portela, 2007) pues el desarrollo de sus procedimientos ("procs") en cuestión de regresión, nos permitirán parametrizar más medidas con las que lograr un modelo mejor ajustado, tanto a nivel de varianza como de media del error. Además, es un programa al que podemos acceder mediante licencia académica de la universidad, y del que los tutores tienen gran dominio.

Este trabajo nace por el interés personal en analizar cómo la información disponible en línea puede servir para predecir en general diversas categorías empresariales, tales como ingresos, número de ventas o el margen de beneficios.

Tras estudiar diferentes alternativas de productos, se decide estimar los ingresos de las películas por dos razones principales. En primer lugar, porque las películas son un producto con bastante movimiento en Internet en general y en las redes sociales en particular. En segundo lugar, porque existe un trabajo afín llevado a cabo por Google (Google. "Quantifying Movie Magic with Google Search." adwords.blogspot.com, June 6, 2013) con el que a través de las búsquedas del tráiler, son capaces de predecir los ingresos de una película. En segundo lugar, porque muchas compañías de Hollywood utilizan Twitter como termómetro para saber el ingreso en taquilla, existiendo diversos estudios al respecto. Aunque siempre midiendo los datos en el día del estreno o incluso una vez estrenada la película. (El País, 2012)

EL programa de Máster en el que se cuadra este trabajo, nos ha enseñado la importancia de incluir múltiples variables para desarrollar modelos de regresión, pues aparte de las relaciones directas podemos llegar a encontrar relaciones indirectas obtenidas por la interrelación de las mismas. Así, haremos un estudio más completo, tomaremos como punto de partida los mencionados datos de Twitter y YouTube ya utilizados en la predicción de ingresos de películas; y los ampliaremos con información proveniente de webs especializadas en películas como son IMDB y BoxOfficeMojo. Paralelamente, nos ayudaremos de *Web Archive*[3], para salvar el problema de la temporalidad y lograr esa información relevante únicamente en la fecha de estreno.

1.3 OBJETIVO O HIPÓTESIS DEL PROYECTO

1.3.1 Objetivo General

El objetivo final del proyecto es conseguir estimar los ingresos que una película comercial tendrá en su primer fin de semana de estreno en taquillas de EEUU, a partir de los datos disponibles hasta 10 días antes de que ese estreno se produzca.

[3] https://archive.org/web/

Por tanto, podemos definir nuestra hipótesis nula como la posibilidad de crear un modelo de regresión a través de variables obtenidas de webs de internet (IMDB y BoxOficeMojo), de YouTube, y de Twitter (Cuenta oficial y término de búsqueda o *hashtag* más popular); con el que estimar 10 días antes del estreno, los ingresos que cualquier película comercial alcanzará en su primer fin de semana de lanzamiento. Todos estos conceptos se explicarán durante el desarrollo del trabajo, a medida que vayan apareciendo en este documento.

1.3.2 Objetivos Específicos.

Podemos detallar ese objetivo amplio, en otros subobjetivos más concretos y abarcables:

- Creación de diferentes modelos de regresión, utilizando múltiples algoritmos, tales como regresión lineales, y machine learning a través de redes neuronales y random forest para la predicción del ingreso de una película en el fin de semana de estreno en EEUU.
- Comparación entre los modelos creados para determinar el más robusto y preciso.
- Identificación de aquellas variables que más influencia tienen en la optimización de esos ingresos, y presentación visual de los resultados.

2 EXPLORACIÓN, ANÁLISIS Y PREPARACIÓN DE LA INFORMACIÓN DE LA BASE DE DATOS.

2.1 LA BASE DE DATOS Y SU CONEXIÓN CON SAS.

Antes de analizar las variables con las que contamos, queríamos realizar una breve descripción de lo que supuso realizar la conexión ODBC de la base de datos y vincularla con SAS para automatizar de cierta forma el flujo de información. Esta es una descripción breve, pues en realidad no forma parte de este proyecto.

El propósito de la capa ODBC (*Open DataBase connectivity* en inglés) es traducir las consultas de datos de la aplicación en comandos que el DBMS entienda. Esto se realiza desde la opción de Herramientas administrativas, ubicada en el panel de control del equipo donde se localiza la base de datos.

Para conectar la base de datos a partir de SQL Server y alimentada con la información de las redes sociales que tenemos preparada, debemos primeramente crear la librería en nuestro entorno SAS, y conectarla a partir de la opción “Engine”, con la opción “ODBC” creada anteriormente. Al presionar “OK”, aparecerá la nueva librería en nuestra lista, y todas las tablas creadas dentro de la BBDD aparecerán como objetos SAS listos para su uso.

2.2 ELECCIÓN DE LAS VARIABLES.

Se realizó una extracción previa de observaciones de las 1.870 películas para analizar solo 425 por motivos de relevancia comercial. Ahora, y como se observará en la descripción de cada variable, debemos realizar un nuevo filtro, por el que trabajaremos los modelos con 399 observaciones. Las otras serán eliminadas por encontrar demasiados missings en las mismas, por ser Outliers o mostrar parámetros de película no comercial.

En lo que respecta a las variables, contamos con 1 variables dependientes que queremos analizar. Por otro lado, contamos con 63 variables independientes. De esas algunas como el ID, el Nombre de la película o la fecha no van a entrar en la elección de los modelos. Necesitamos analizar las 60 variables restantes antes de proceder a desarrollar los modelos, con esto se mejorará la predictibilidad de las mismas.

Para realizar este análisis, hemos utilizado el programa Enterprise Miner (eMiner) de SAS, pues ofrece de manera dinámica un estudio amplio para conocer las características de cada variable de forma sencilla a través del nodo Multitrazado ('MultiPlot'). (SAS Institute, 2013)

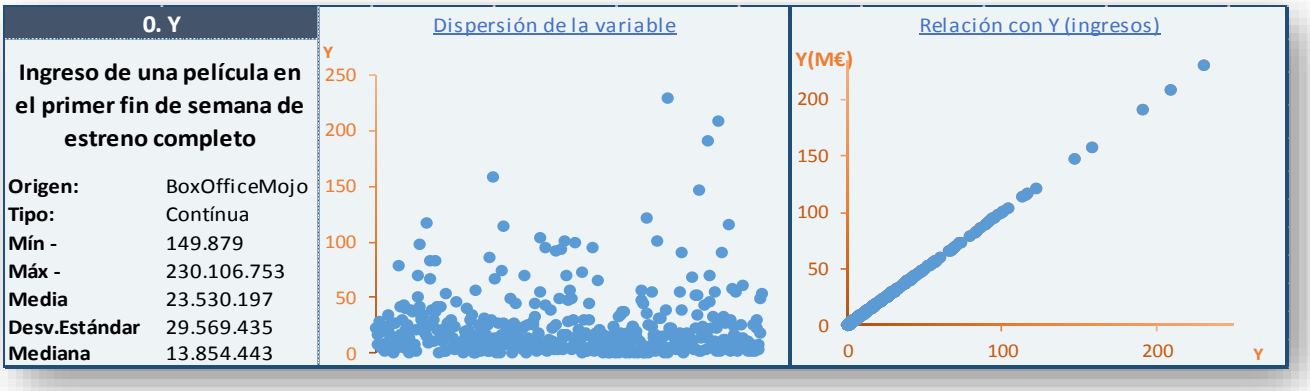
Además, como hemos explicado, la modelización de modelos se realizará en con el programa SAS 9.4, y el tipo de fichero generado por este código es compatible con eMiner. Para mejorar la presencia de eMiner, utilizamos Excel.

La información incluye algunos datos de distribución a la derecha; así como una gráfica de distribución en el centro con la distribución de la variable (eje Y), según las 425 películas que analizamos (eje X); y finalmente una gráfica de relación de cada variable con los ingresos a la derecha, correspondiendo ingresos (eje Y), y la variable independiente (eje X). Ademas, comentaremos aquellas sobre las que hayamos realizado algún tipo de tratamiento externo para mejorar su usabilidad.

2.3 ANÁLISIS DE LA VARIABLE DEPENDIENTE (Y).

Se ha extraído la información de dos bases de datos diferentes para poder comparar y crear un óptimo y evitar errores como ya explicamos. Ambas variables aportaban el mismo dato en 350 observaciones; y además de las 75 restantes, las diferencias eran menores al 20% en otras 73 que solucionamos promediando los resultados. Las únicas 3 observaciones que aportaban datos con mayores diferencias, las solucionamos buscándolas manualmente para corroborar qué dato es el más fiable.

Imagen 1: Explicación de la variable dependiente

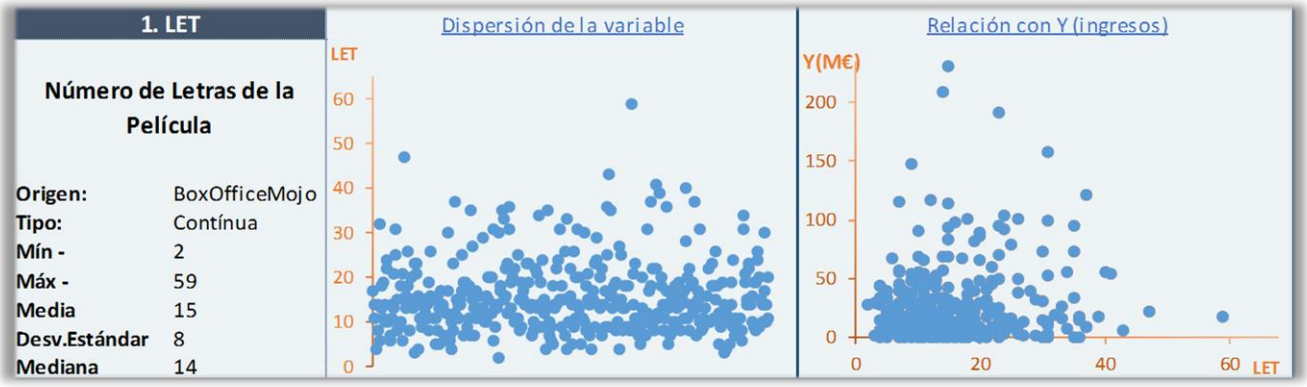


Posteriormente probaremos a la hora de realizar los modelos el logaritmo neperiano LNY, pues al tratarse de una variable de ingresos con gran variabilidad, es posible que se comporte mejor; y sea más fácilmente predecible.

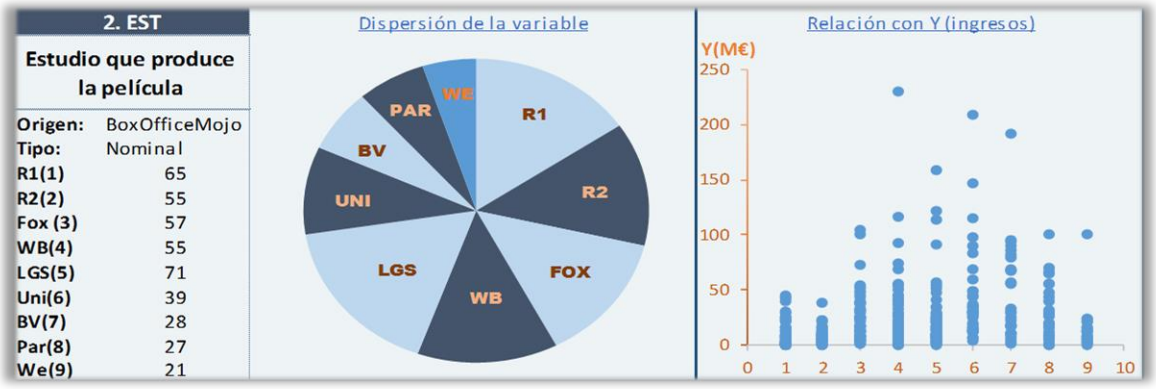
2.4 ANÁLISIS DE LAS VARIABLES INDEPENDIENTES.

LET analiza si existe alguna correlación entre el nombre de la película y su capacidad de generar ingresos. En este sentido, se va a analizar si el número de letras de una película puede influir; es decir, si la complejidad de los nombres puede ser un hándicap en la recaudación o no.

Imagen 2: Explicación secuencial de las variables independientes

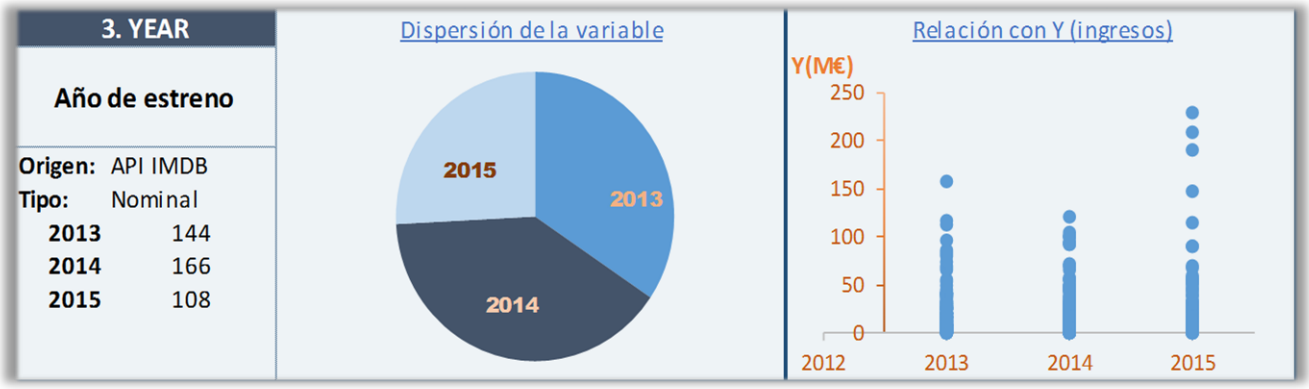


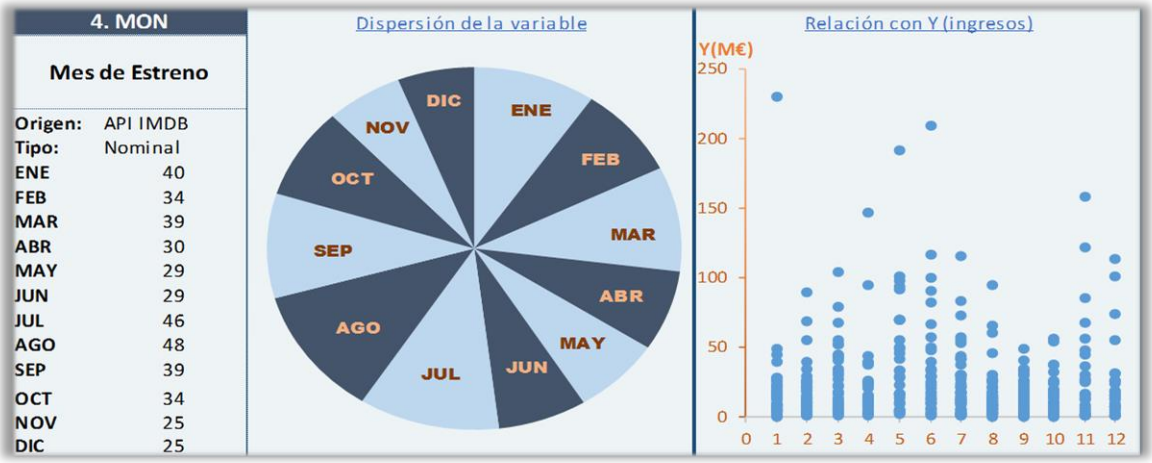
EST trata el estudio que ha realizado la película. Al analizar la base completa había 45 diferentes productoras. Para disminuir el número de categorías, realizamos 3 acciones: En primer observamos que Sony y Lionsgate realizan muchas películas en conjunto, lo que unido a las noticias que, desde 2014, hablan de una posible unión entre ellas, nos lleva a decidir tratarlas como una única nueva categoría; en segundo lugar unimos un conjunto de 28 pequeñas productoras, que en los tres años de análisis habían realizado menos de 10 películas(R1); por último unimos 4 productoras que apenas han realizado entre 10 y 20 películas entre 2013 y 2015(R2). Con ello, tenemos 9 categorías.



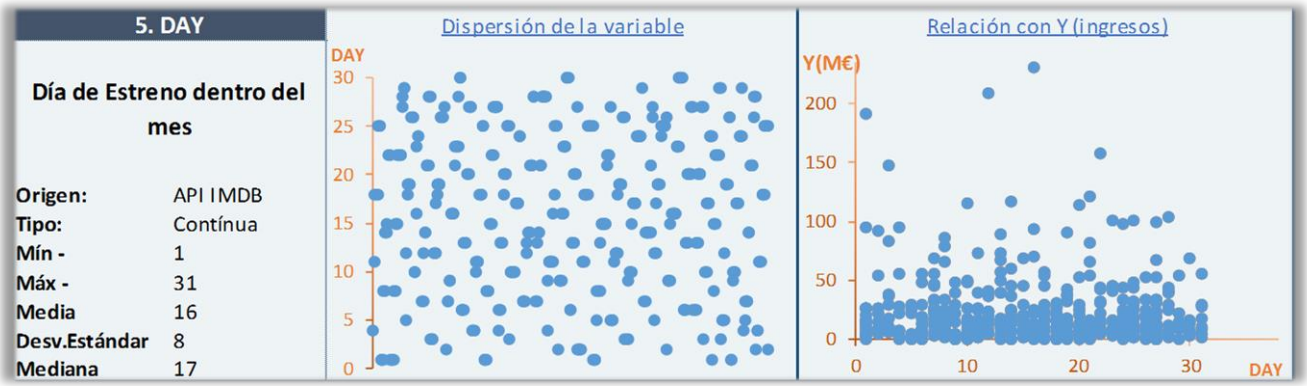
La variable de Fecha (**DATA**) es una de las más importantes, sobretodo de cara a obtener las variables que pueden variar con el tiempo. Aunque esta variable no se incorporara al modelo como tal, si analizaremos otras provenientes de esta misma como son:

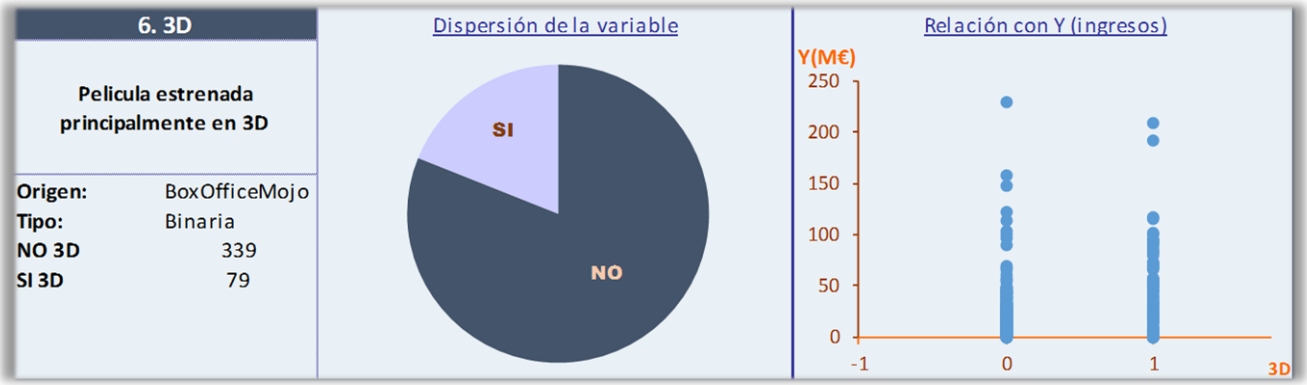
YEAR. Películas estrenadas entre 2013 y 2015. Dos motivos principales nos llevan a estudiar esta variable. Por un lado, la evolución de la crisis económica en EEUU puede afectar en la recaudación de las películas, sobretodo penalizando los ingresos de 2013. Y por otro lado, la rápida evolución de las redes sociales, puede llevar a que los patrones de 2013 sean diferentes a los de 2014 o 2015. Se pretende por tanto controlar la ocurrencia de estas dos posibilidades, incluyendo esta variable.



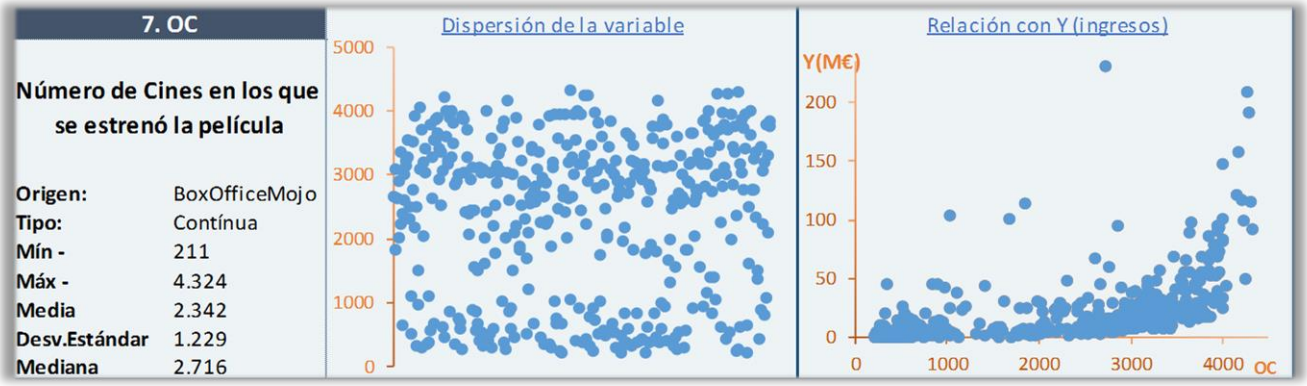


DAY. Incluimos el día del mes en el que se estrena, pero no la variable que distingue los días según qué día de la semana se trate, pues el 84% de las películas se estrenan en Viernes, y el 10% en Jueves; dejando una variabilidad limitada.

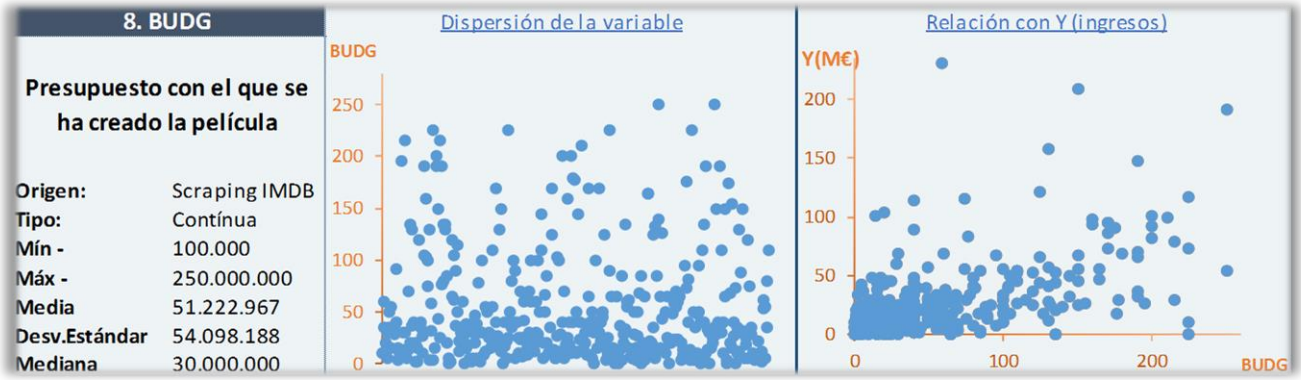




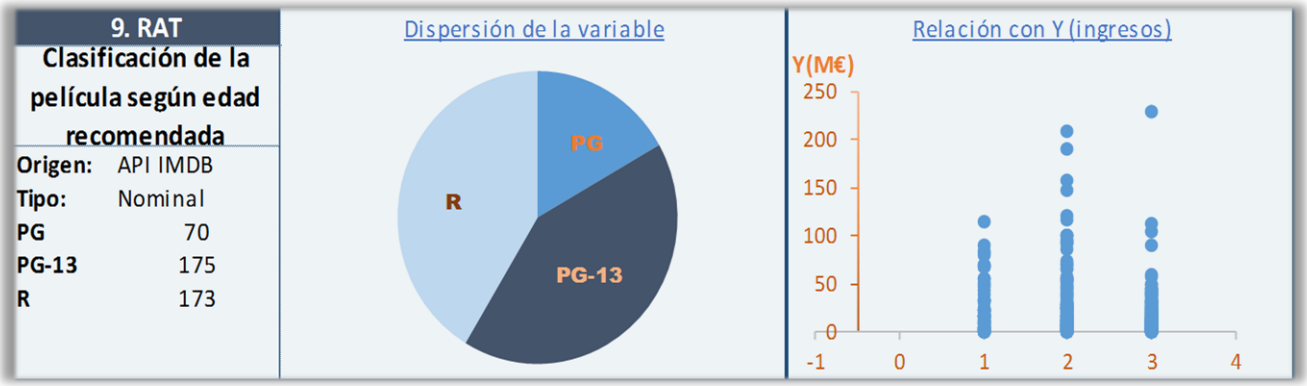
OC. Como ya indicamos antes, para nuestro análisis es fundamental que esta variable cumpla con unos mínimos, pues si el número de cines es demasiado bajo, influye en de una manera demasiado directa en la recaudación.



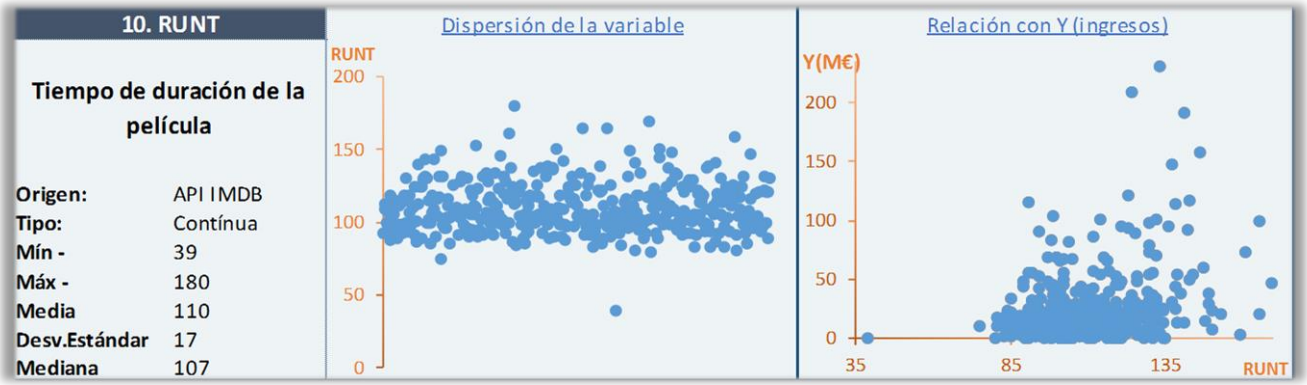
BUDG. Algunas de las películas no liberan el presupuesto de su película. Así una vez terminada la extracción mediante web scraping que explicamos anteriormente, nos encontramos con 30 observaciones missing, de las que conseguimos información manual de 9 de ellas. Las restantes 21 las imputamos según el resto de variables con eMiner ('Nodo Imputación').



RAT. En EEUU nos encontramos con 5 clasificaciones diferentes en función de la recomendación de la película: G (todos los públicos), PG (no apta para menores de 10 salvo acompañados por un adulto), PG-13 (no apta para menores de 13 salvo acompañados por un adulto), R (no apta para menores de 17 salvo acompañados por un adulto), PG-17 (solo apta para mayores de 18 años).(“Clasificación por edades (cine),” 2015). En nuestro caso PG-17 no aparece en ninguna película. Además, unimos G y PG.

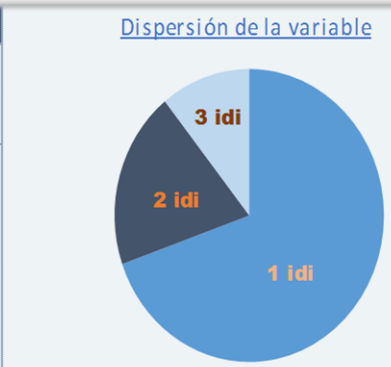


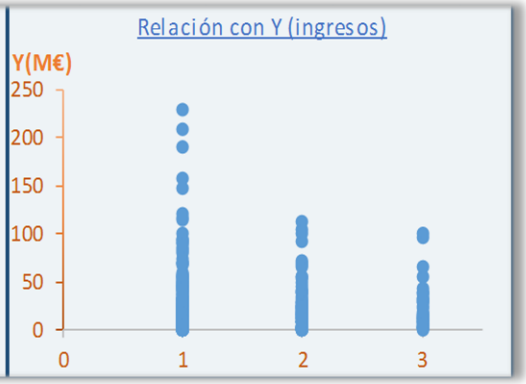
RUNT. Eliminamos una observación pues se trata de un corto de apenas 39 minutos.



L_N. Las películas contaban con hasta 5 idiomas, unimos en una solo variable más de 3 idiomas.

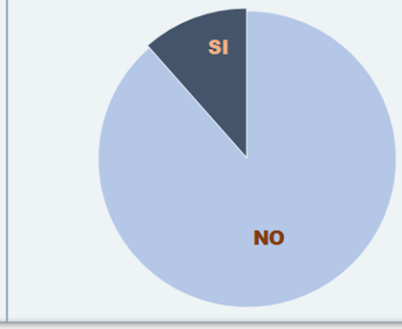
11. L_N	
Número de Idiomas en los que se ha grabado la película	
Origen:	API IMDB
Tipo:	Contínua
1 idioma	291
2 idiomas	82
3 idiomas	45

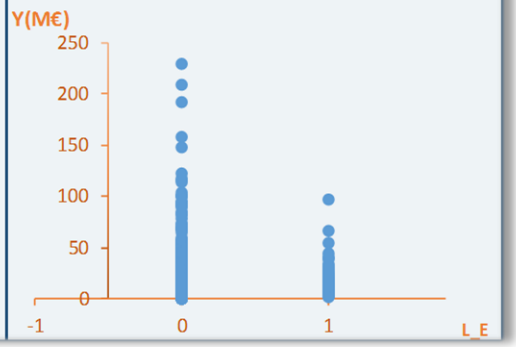




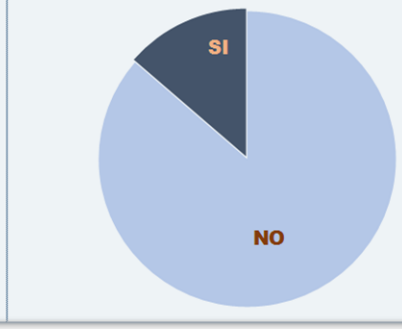
Contamos con unos 50 idiomas (44 aparecen en apenas una o dos películas de manera aislada por lo que los obviamos). De las seis restantes analizamos cinco de ellos (español, ruso, francés, alemán y japonés); el español en solitario pues es el más utilizado después del inglés, y el resto en conjunto. En cuanto al inglés, no lo analizamos pues aparece en el 98,5% de las películas, y por tanto no sirve como variable predictiva.

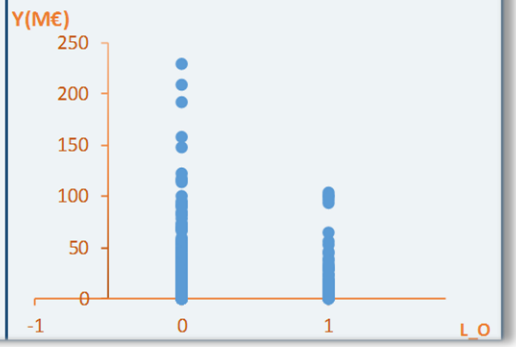
12. L_E	
Incluye idioma Español	
Origen:	API IMDB
Tipo:	Binaria
NO	370
SI	48



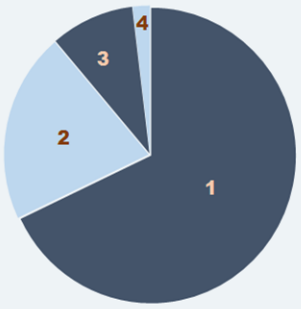
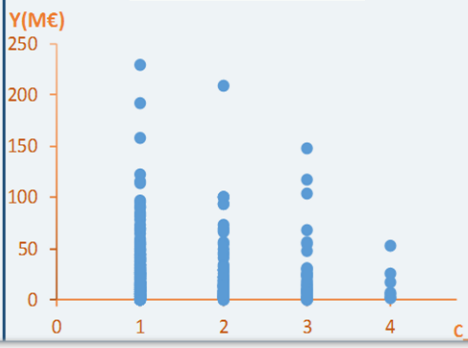


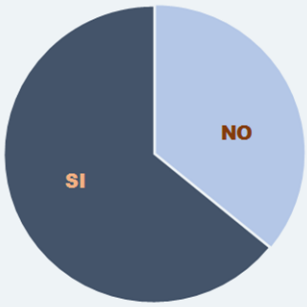
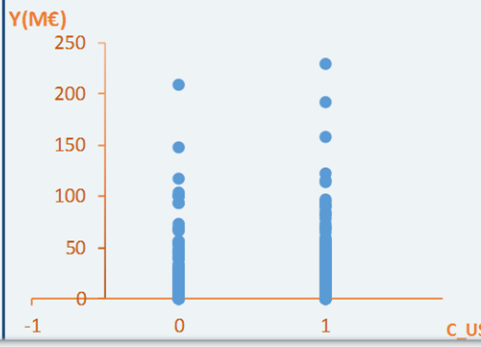
13. L_O	
Incluye otro idioma principal (Ruso, Alemán, Francés o Japones)	
Origen:	API IMDB
Tipo:	Binaria
NO	361
SI	57

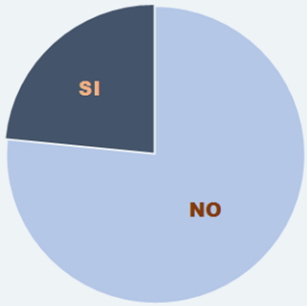
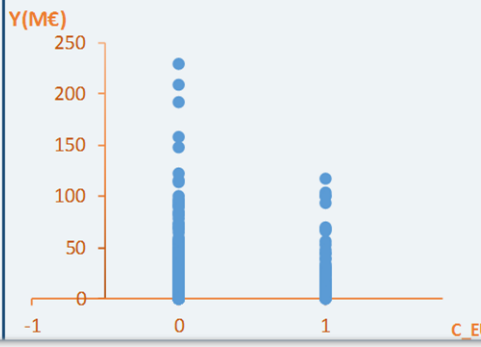


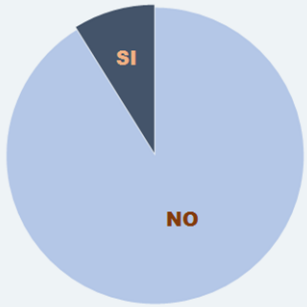
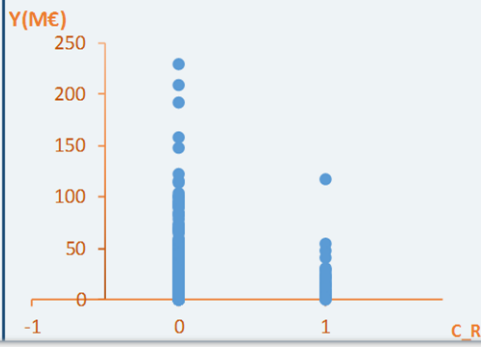


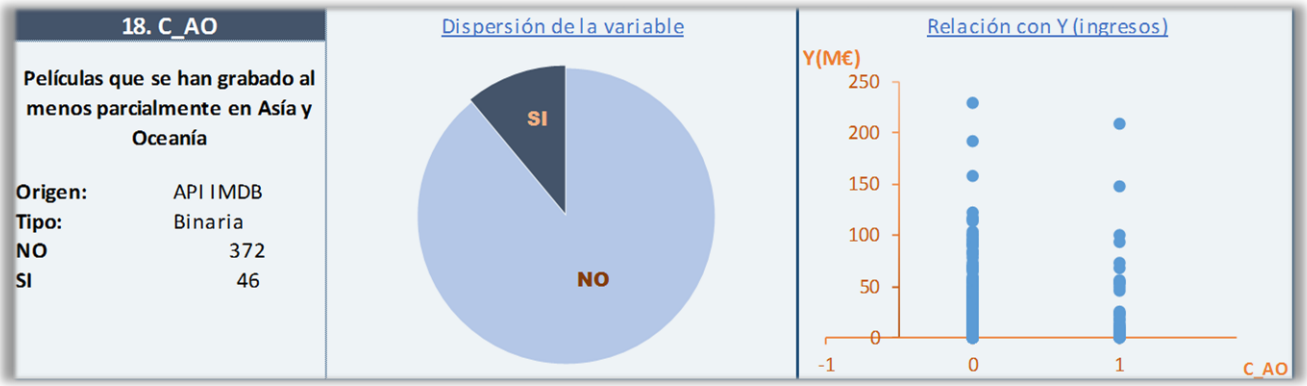
En cuanto al país de grabación, encontramos que una película puede grabarse hasta en 6 países diferentes. Creamos una variable que identifique cuantos países ha sido, uniendo más de 4 en una sola categoría debido a su baja frecuencia. Además, dividimos por tanta esta variable en varias dummies en función de si se ha grabado únicamente en EEUU, o si tiene presencia en cada uno de los continentes.

14. C_N		Dispersión de la variable	Relación con Y (ingresos)
Número de países en los que se ha grabado la película			
Origen:	API IMDB		
Tipo:	Contínua		
1 país	284		
2 paises	87		
3 paises	39		
4 paises o más	8		

15. C_USA		Dispersión de la variable	Relación con Y (ingresos)
Películas que se han grabado exclusivamente en EEUU			
Origen:	API IMDB		
Tipo:	Binaria		
NO	150		
SI	268		

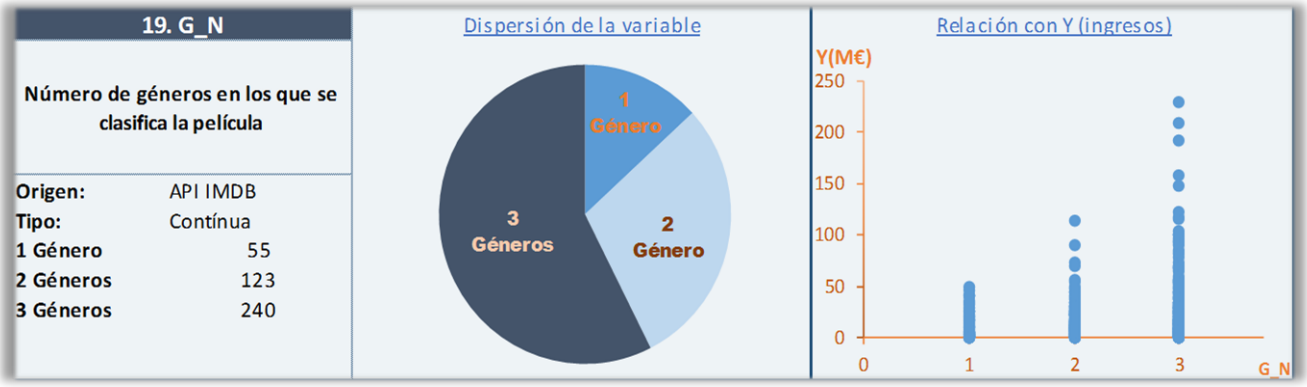
16. C_EU		Dispersión de la variable	Relación con Y (ingresos)
Películas que se han grabado al menos parcialmente en Europa			
Origen:	API IMDB		
Tipo:	Binaria		
NO	320		
SI	98		

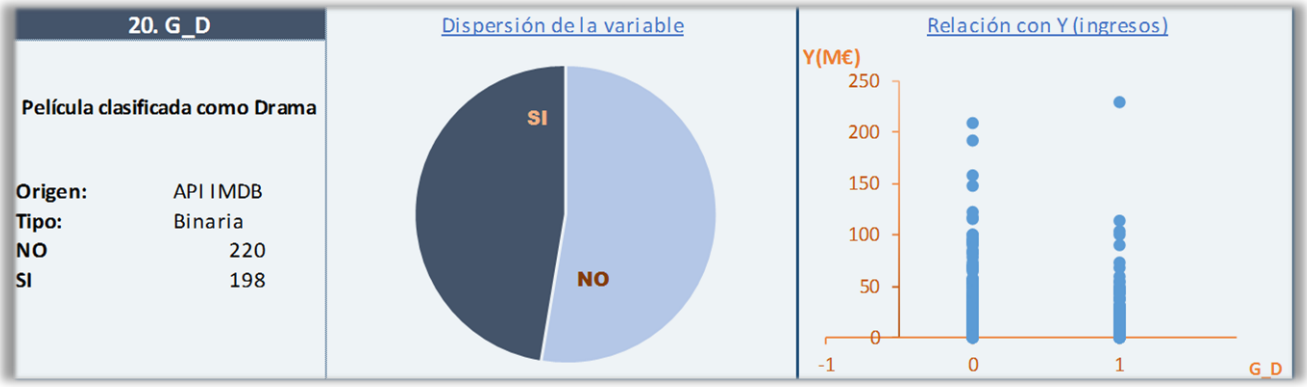
17. C_RA		Dispersión de la variable	Relación con Y (ingresos)
Películas que se han grabado al menos parcialmente en el América (sin EEUU)			
Origen:	API IMDB		
Tipo:	Binaria		
NO	381		
SI	37		

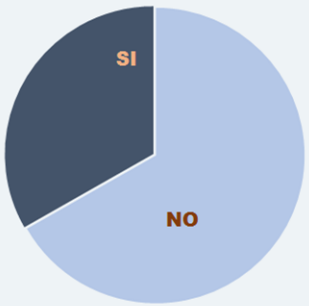
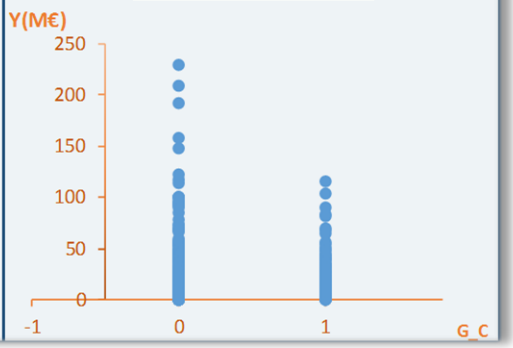


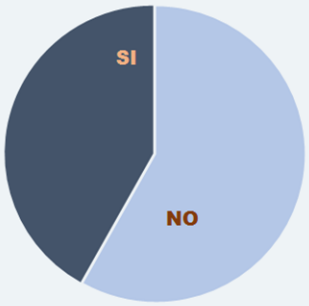
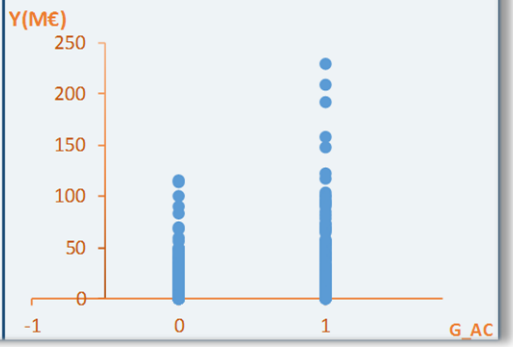
G_N. La mayoría de películas se graban con 3 géneros. Y existe correlación entre colocar más géneros en la clasificación y obtener más ingresos.

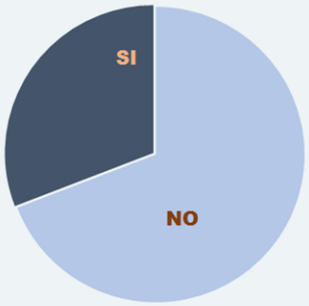
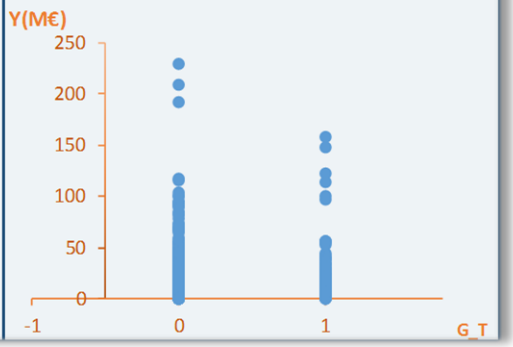
Existen 22 géneros en los que se clasifican nuestra base de datos. Realizamos una unión de géneros que tengan cierta similitud en su definición, llegando a tener 11 géneros. De esos, estudiamos 10 de manera independiente como variables binarias. Uno de ellos, Deporte, no lo estudiamos porque su frecuencia es apenas de 11. Por tanto, el estudio de las próximas 10 variables debe realizarse de manera conjunta, para obtener mejores resultados.

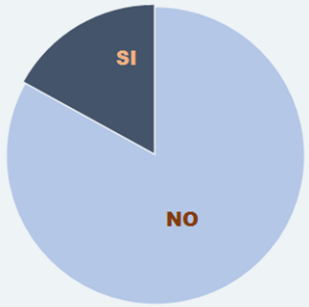
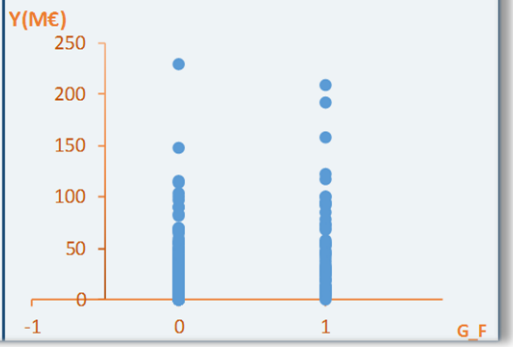


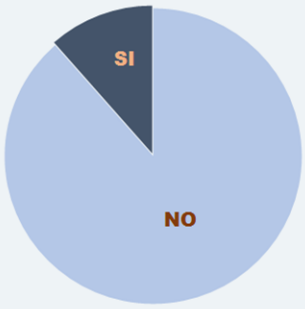
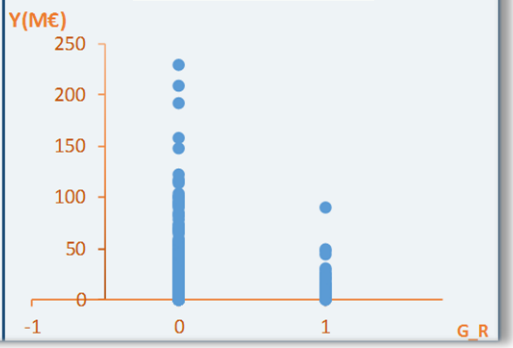


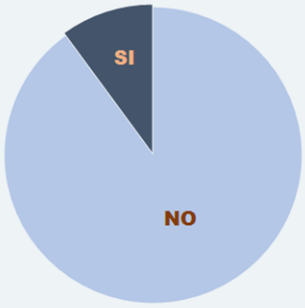
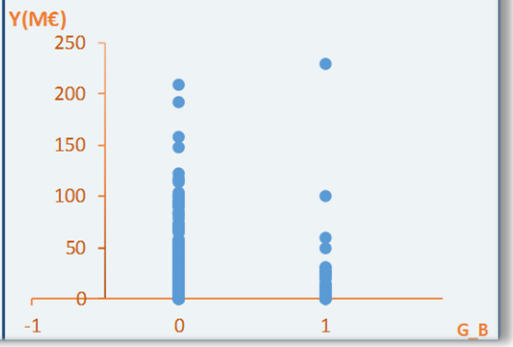
21. G_C	Dispersión de la variable	Relación con Y (ingresos)
Película clasificada como Comedia **Origen:** API IMDB **Tipo:** Binaria **NO** 279 **SI** 139		

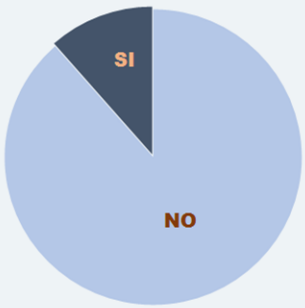
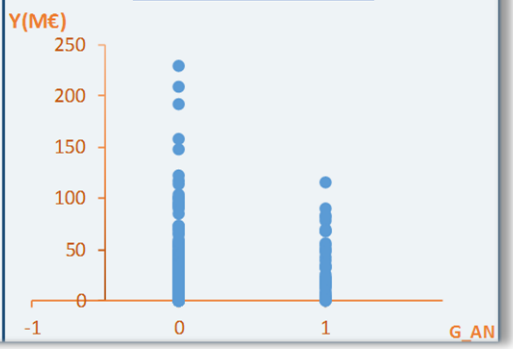
22. G_AC	Dispersión de la variable	Relación con Y (ingresos)
Película clasificada como Acción **Origen:** API IMDB **Tipo:** Binaria **NO** 243 **SI** 175		

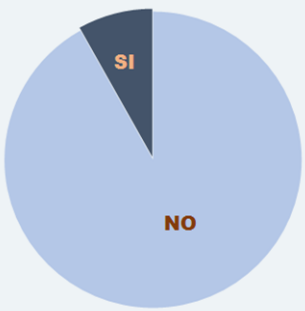
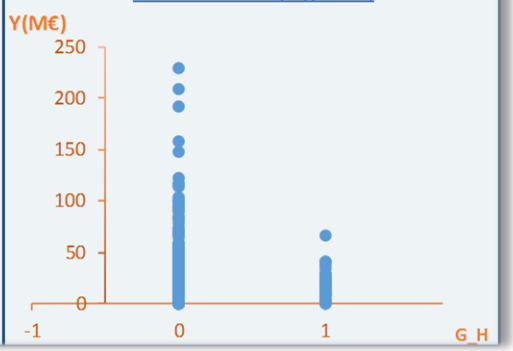
23. G_T	Dispersión de la variable	Relación con Y (ingresos)
Película clasificada como Thriller **Origen:** API IMDB **Tipo:** Binaria **NO** 289 **SI** 129		

24. G_F	Dispersión de la variable	Relación con Y (ingresos)
Película clasificada como Fantasía o Sci-Fi **Origen:** API IMDB **Tipo:** Binaria **NO** 347 **SI** 71		

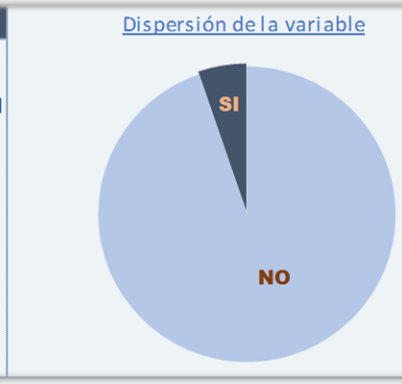
25. G_R	Dispersión de la variable	Relación con Y (ingresos)
Película clasificada como Romance **Origen:** API IMDB **Tipo:** Binaria **NO** 370 **SI** 48		

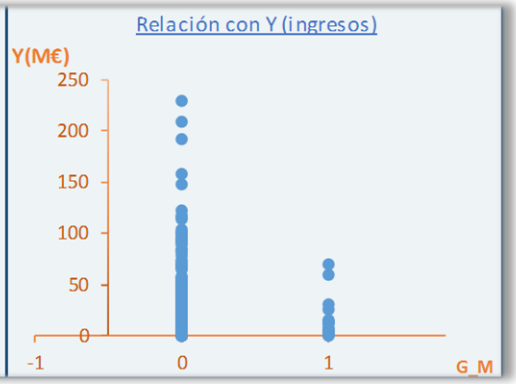
26. G_B	Dispersión de la variable	Relación con Y (ingresos)
Película clasificada como Biografía **Origen:** API IMDB **Tipo:** Binaria **NO** 376 **SI** 42		

27. G_AN	Dispersión de la variable	Relación con Y (ingresos)
Película clasificada como Animación **Origen:** API IMDB **Tipo:** Binaria **NO** 370 **SI** 48		

28. G_H	Dispersión de la variable	Relación con Y (ingresos)
Película clasificada como Horror **Origen:** API IMDB **Tipo:** Binaria **NO** 384 **SI** 34		

29. G_M	
Película clasificada como Musical	
Origen:	API IMDB
Tipo:	Binaria
NO	396
SI	22

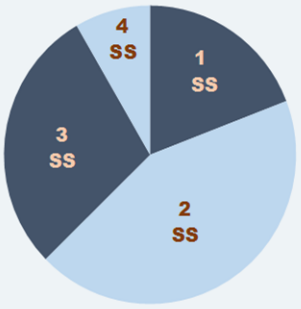


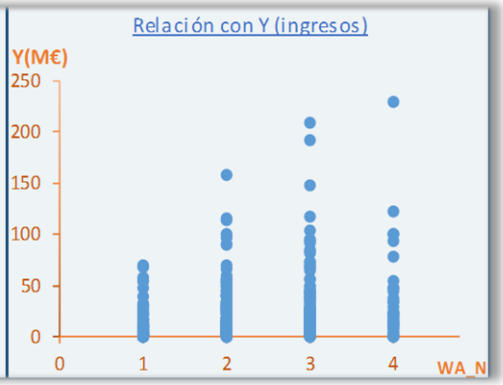


Cuando descargamos la información de Web Archive, obtuvimos la información de 4 fechas diferentes pues pensábamos analizarla entre 1 mes y 10 días antes del estreno. Sin embargo descartamos esta opción debido a la baja variabilidad o incluso a que antes de 10 días la información es demasiado limitada. Por ello utilizamos las variables únicamente de la fecha más cercana al estreno.

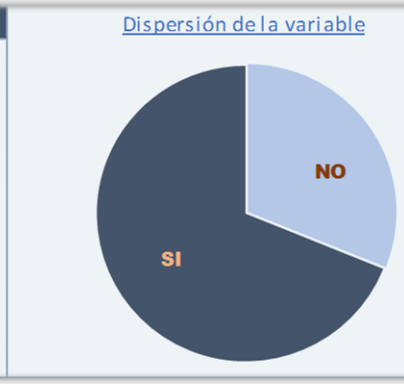
Hay que destacar que ciertas películas deciden no abrir las votaciones hasta después de estrenar la película. En ese sentido creamos una variable binaria que nos indique si se abrió o no la posibilidad de votar 10 días antes del estreno. Y colocamos el valor 0 en las variables que no han abierto las votaciones para el resto de variables, pues entendemos que no se tratan de valores missing a estimar, sino que se trata de una decisión que conocemos, y por tanto queremos ver cómo se comporta en ese sentido.

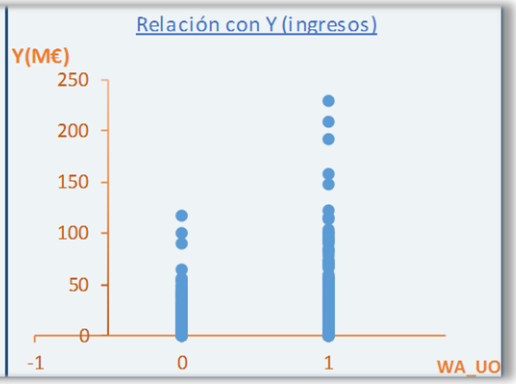
30. WA_N	
Fechas diferentes que hemos encontrado en ScreenShots en WA en torno a 10 días antes del estreno	
Origen:	API WA - IMDB
Tipo:	Nominal
1 ScreenShot	80
2 ScreenShot	182
3 ScreenShot	121
4 ScreenShot	35



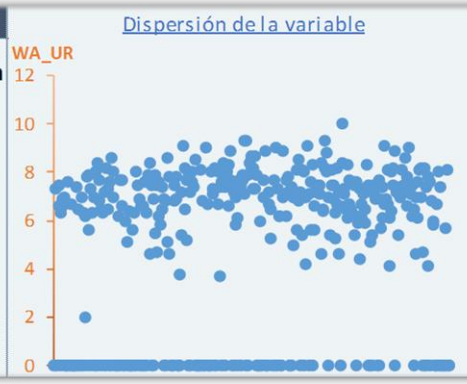


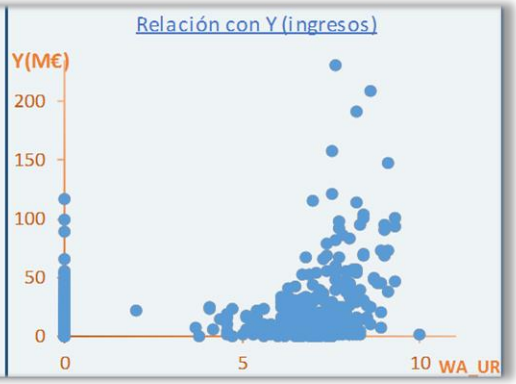
31. WA_UO	
Posibilidad del Usuario para votar en IMDB en torno a 10 días antes del estreno	
Origen:	API WA - IMDB
Tipo:	Binaria
NO	130
SI	288





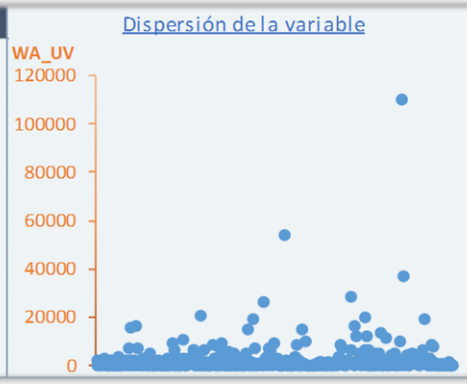
32. WA_UR	
Rating de los Usuarios de 1 a 10 en torno a 10 días antes del estreno	
Origen:	API WA - IMDB
Tipo:	Contínua
Mín -	0
Máx -	10
Media	5
Desv.Estándar	3
Mediana	7

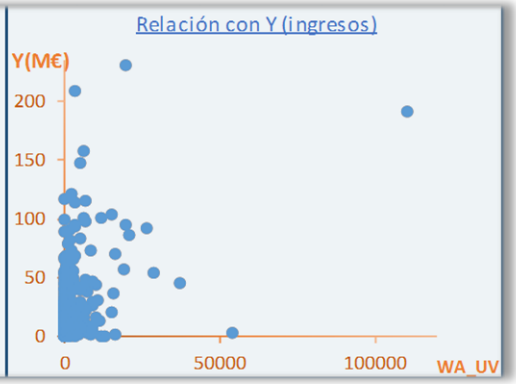




WA_UV. Se observan Outliers que debemos arreglar antes de realizar los modelos.

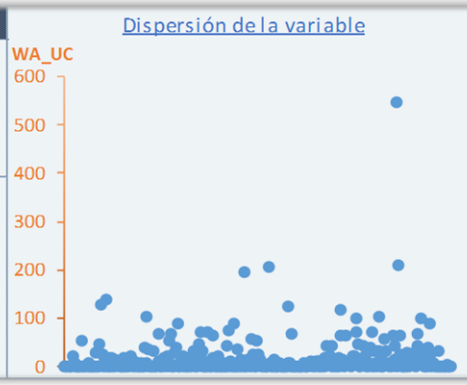
33. WA_UV	
Número de Usuarios que han votado en torno a 10 días antes del estreno	
Origen:	API WA - IMDB
Tipo:	Contínua
Mín -	0
Máx -	110.516
Media	2.417
Desv.Estándar	7.189
Mediana	573

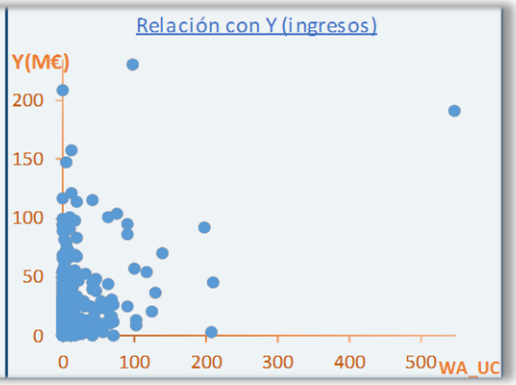




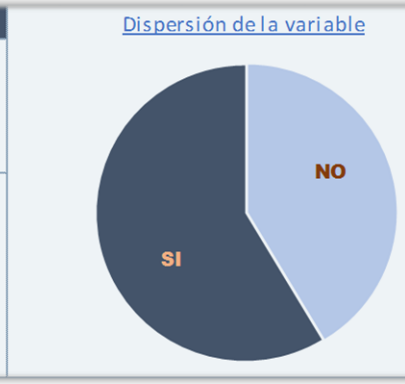
WA_UC. Se observan Outliers que debemos arreglar antes de realizar los modelos.

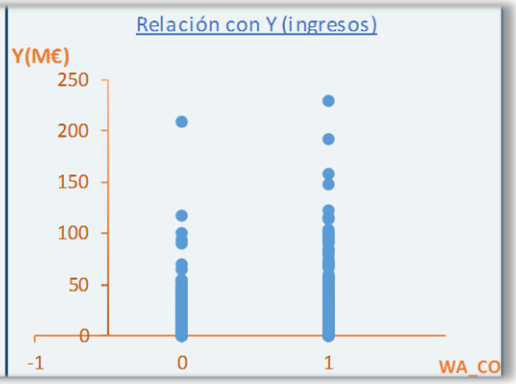
34. WA_UC	
Número de Críticas que han realizado los Usuarios en torno a 10 días antes del estreno	
Origen:	API WA - IMDB
Tipo:	Contínua
Mín -	0
Máx -	547
Media	15
Desv.Estándar	38
Mediana	0





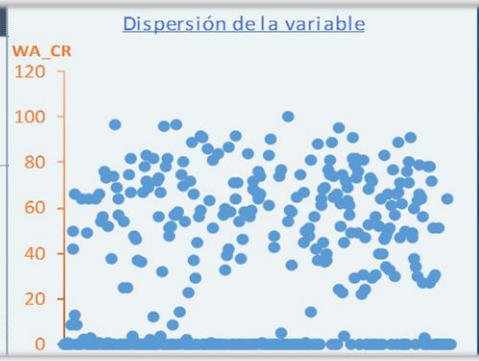
35. WA_CO	
Posibilidad de los Críticos para votar en IMDB en torno a 10 días antes del estreno	
Origen:	API WA - IMDB
Tipo:	Binaria
NO	173
SI	245

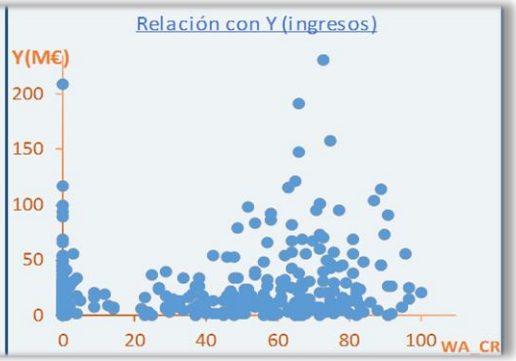




WA_CR. Esto mide el MetaScore de IMDB, una votación de críticos expertos.

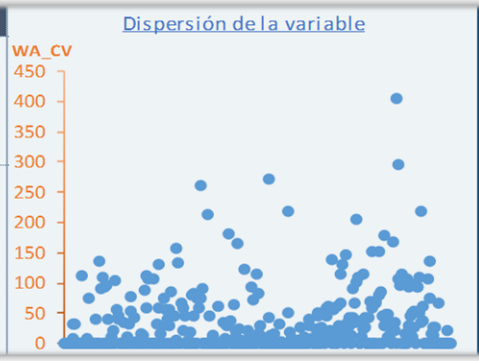
36. WA_CR	
Rating de los Críticos de 1 a 100 en torno a 10 días antes del estreno	
Origen:	API WA - IMDB
Tipo:	Contínua
Mín -	0
Máx -	100
Media	32
Desv.Estándar	33
Mediana	27

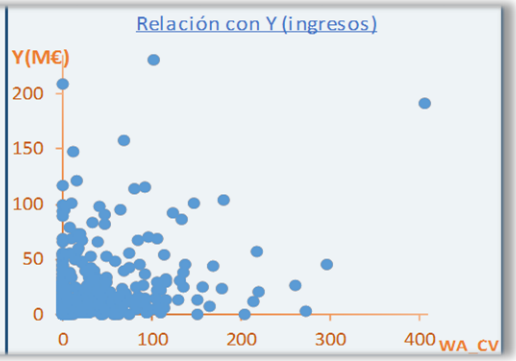




WA_CV. De nuevo encontramos datos Outliers que debemos arreglar.

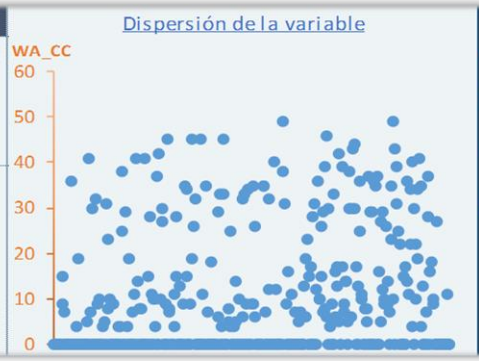
37. WA_CV	
Número de Críticos que han votado en torno a 10 días antes del estreno	
Origen:	API WA - IMDB
Tipo:	Contínua
Mín -	0
Máx -	406
Media	31
Desv.Estándar	51
Mediana	2

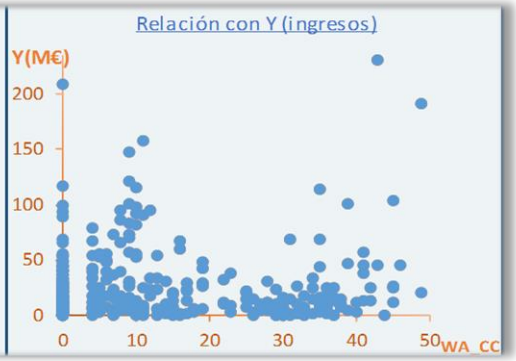




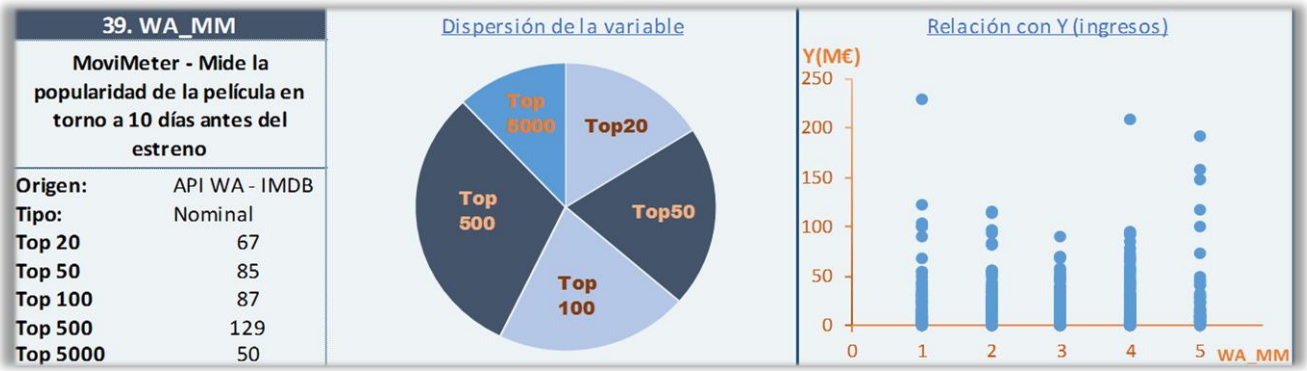
WA_CC. Existen dato Outliers que debemos arreglar antes de la creación de modelos.

38. WA_CC	
Número de Críticas que han realizado los críticos en torno a 10 días antes del estreno	
Origen:	API WA - IMDB
Tipo:	Contínua
Mín -	0
Máx -	49
Media	10
Desv.Estándar	14
Mediana	4

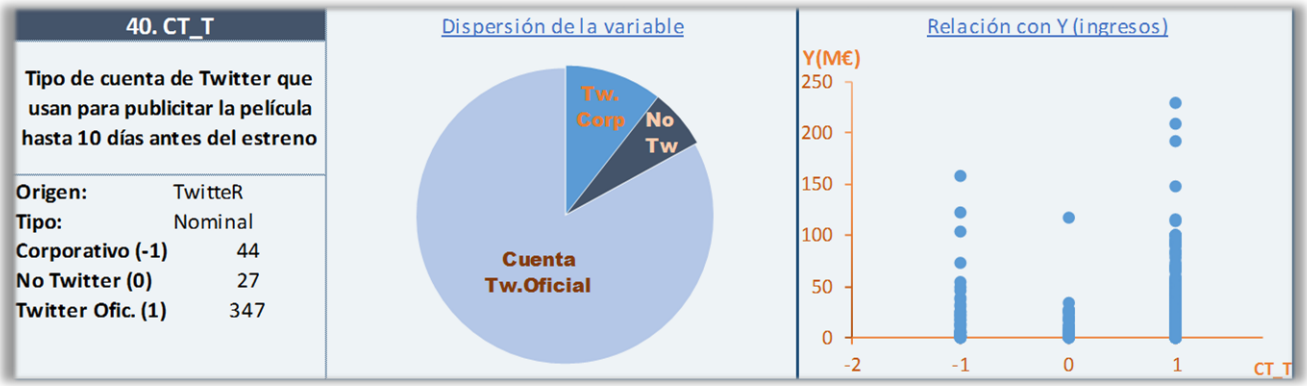


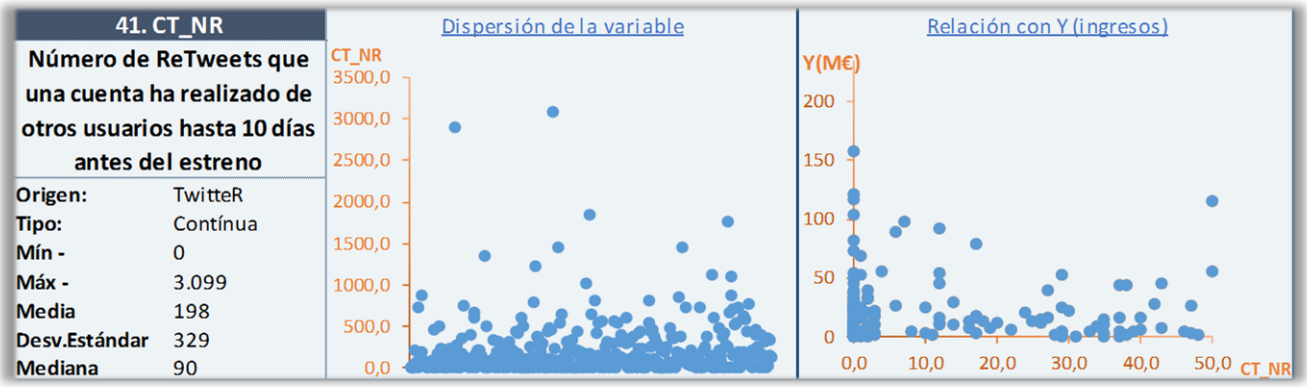


WA_MM. En un principio se clasificaba de 1 a 100, y si no estaba entre los 100 primeros, la clasificaba en rangos. Para normalizar toda la información decidimos transformarla en una variable nominal, haciendo rangos de todas las películas.



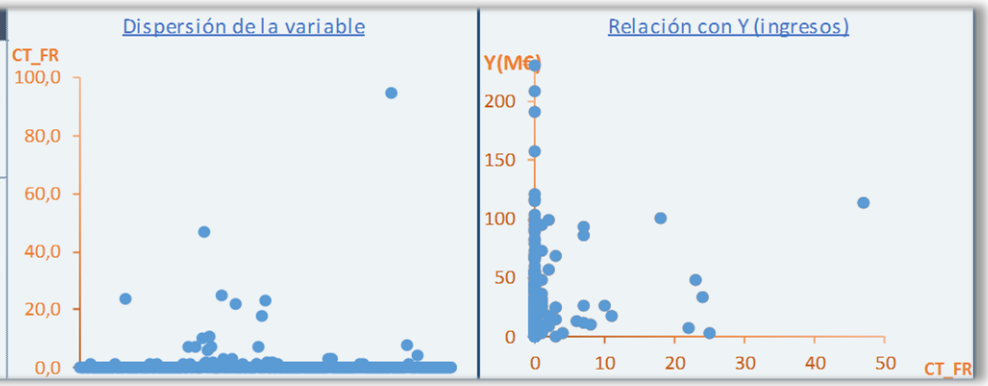
Pasamos al siguiente conjunto de variables, las creadas a partir de la cuenta oficial de Twitter. Las cuentas oficiales pueden realizar dos tipos de actividades, por un lado pueden crear escribir Tweets, y por otro lado pueden retwitter mensajes de otras personas. De cada uno de esos grupos obtendremos el número de tweets, el número de veces que se le ha dado favorito, y el número de retweets que se han realizado.





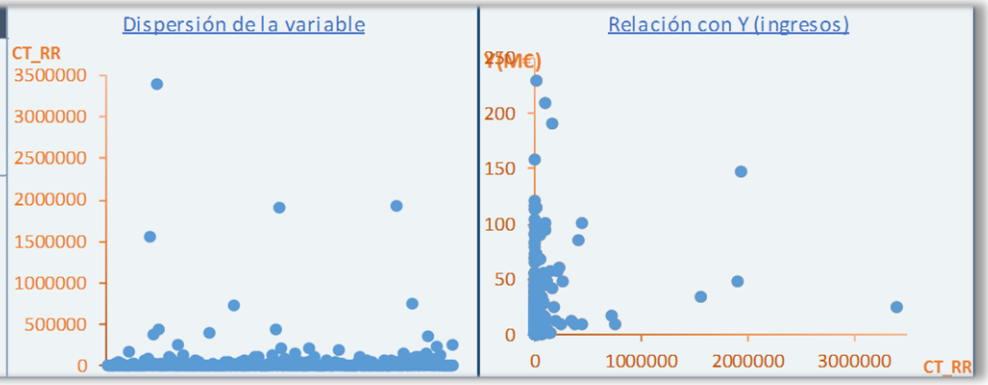
CT_FR. De nuevo debemos arreglar los Outliers antes de la creación de modelos.

42. CT_FR	
Número de veces que esos ReTweets han recibido favorito hasta 10 días antes del estreno	
Origen:	TwitteR
Tipo:	Contínua
Mín -	0
Máx -	95
Media	1
Desv.Estándar	6
Mediana	0

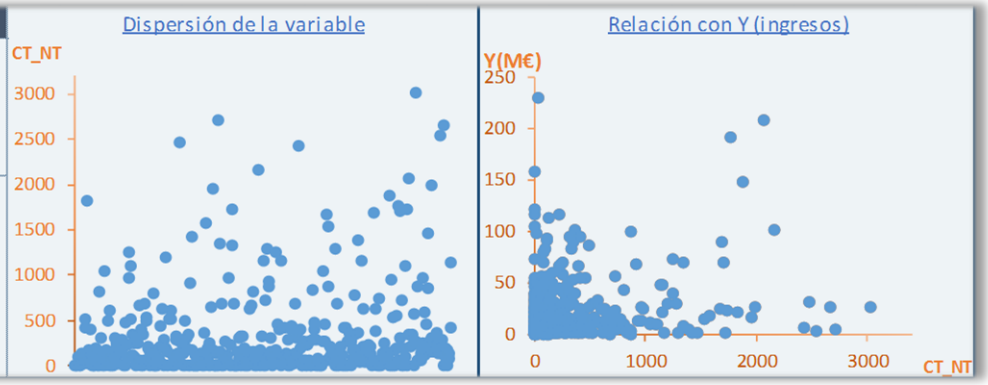


CT_RR. De nuevo debemos arreglar los Outliers antes de la creación de modelos.

43. CT_RR	
Número de veces que esos ReTweets se han seguido "retweeteando" hasta 10 días antes del estreno	
Origen:	TwitteR
Tipo:	Contínua
Mín -	0
Máx -	3.395.930
Media	46.840
Desv.Estándar	235.105
Mediana	3.862

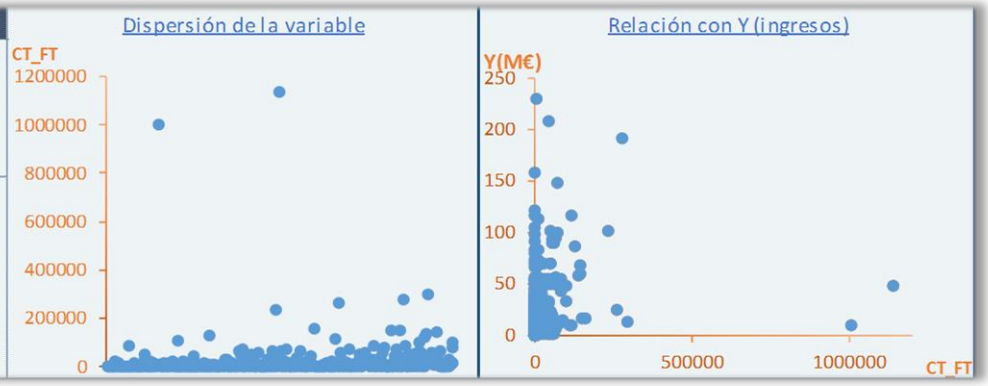


44. CT_NT	
Número de Tweets que una cuenta ha escrito hasta 10 días antes del estreno	
Origen:	TwitteR
Tipo:	Contínua
Mín -	0
Máx -	3.018
Media	327
Desv.Estándar	499
Mediana	146

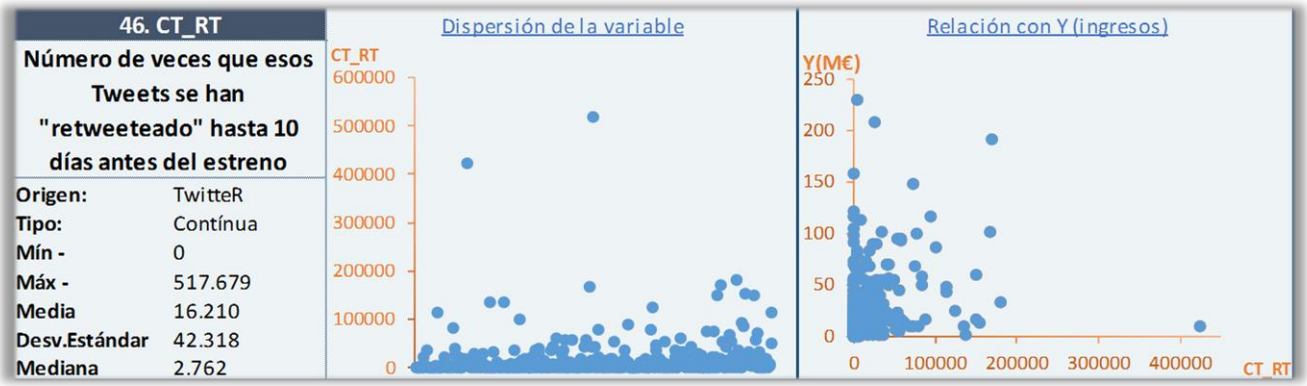


CT_FT. De nuevo debemos arreglar los Outliers antes de la creación de modelos.

45. CT_FT	
Número de veces que esos Tweets han recibido favorito hasta 10 días antes del estreno	
Origen:	TwitteR
Tipo:	Contínua
Mín -	0
Máx -	1.137.282
Media	21.842
Desv.Estándar	81.470
Mediana	3.020

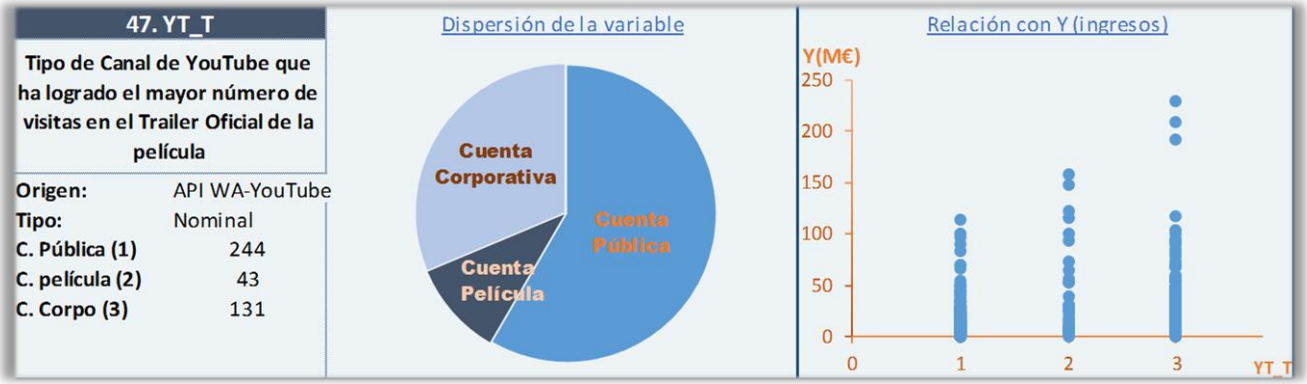


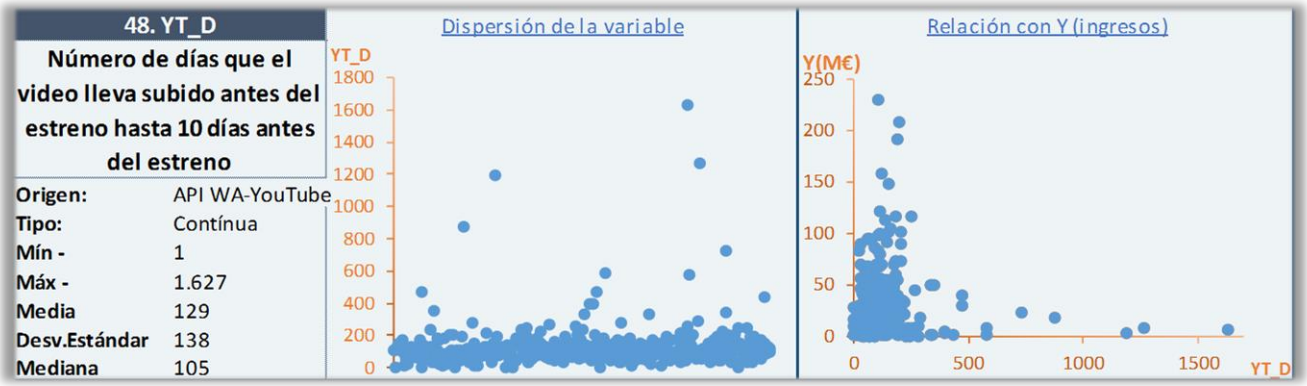
CT_RT. De nuevo debemos arreglar los Outliers antes de la creación de modelos.



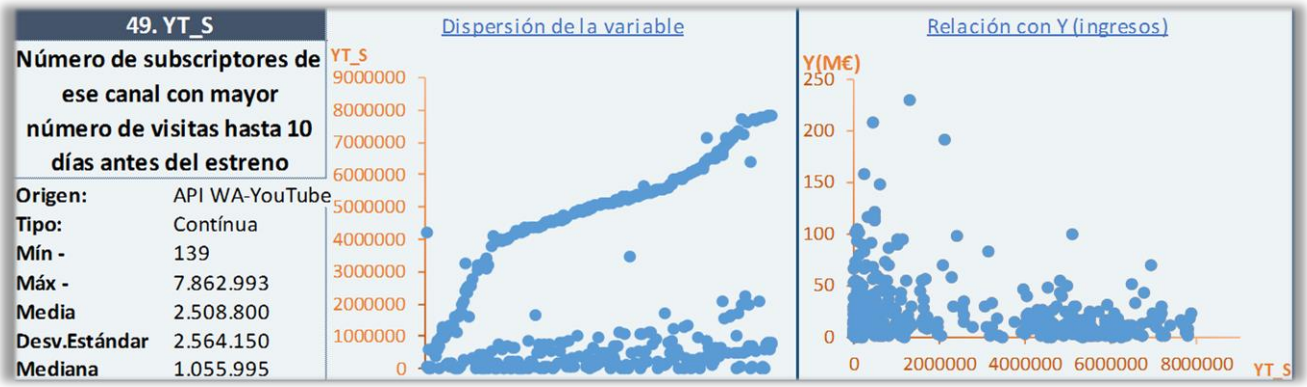
El siguiente grupo de variables han sido obtenidas de YouTube, para este análisis escogimos dentro del tráiler oficial, aquel video que tuviera más visitas, pues encontramos que normalmente uno de los videos obtenía sustancialmente más visualizaciones que el resto; y no siempre tiene por qué ser la cuenta oficial. De hecho normalmente se trata de una cuenta llamada MoviClip.

YT_T. Medimos si el tipo de canal afecta a los ingresos. Como se observa normalmente es un canal público el que tiene más visitas. Y cuando esto ocurre, los ingresos se mantienen menores que en cuentas propias de la película o aquellas que utilizan el canal oficial del estudio.

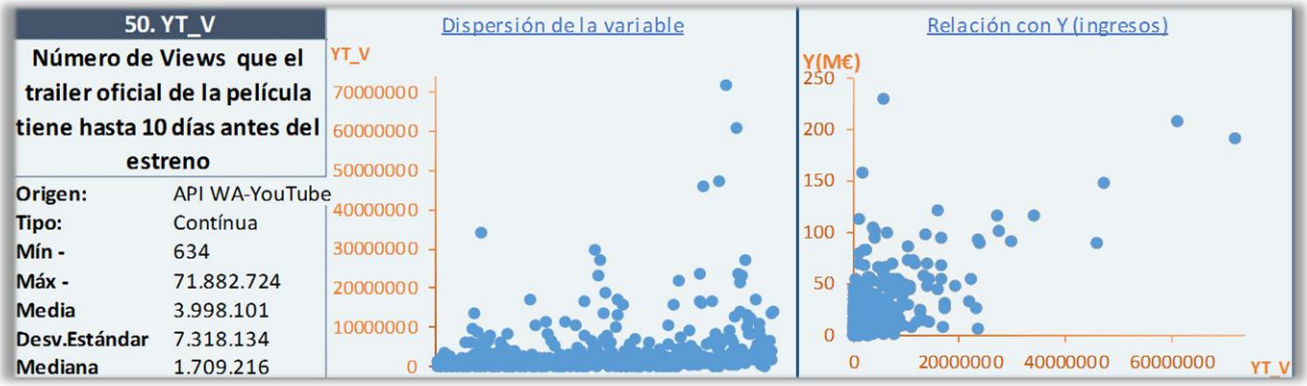




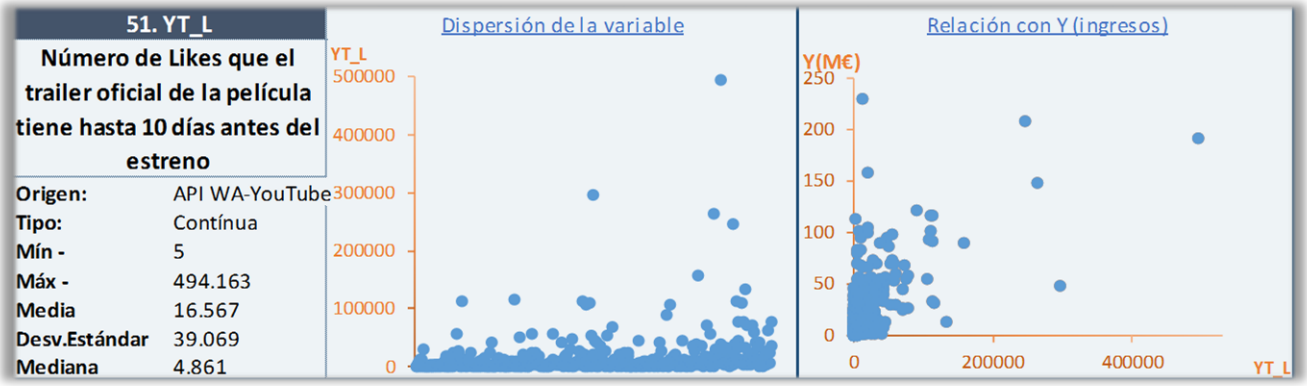
YT_D. El número de usuarios presenta una curva destacable en la serie de MovieClip, debido a que las películas están ordenadas por fecha de estreno, y cada vez que pasa más tiempo el canal tiene más suscritores; aparte de eso no se observan otras relaciones directas.



YT_V. Debemos arreglar los Outliers antes de realizar los modelos de estimación.

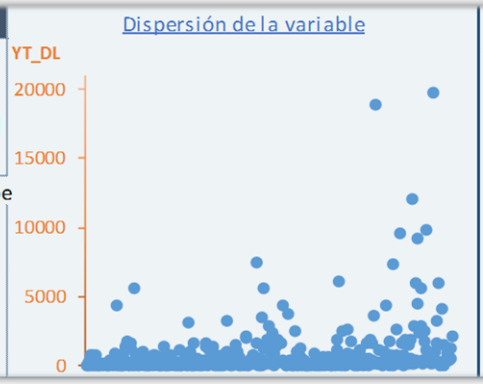


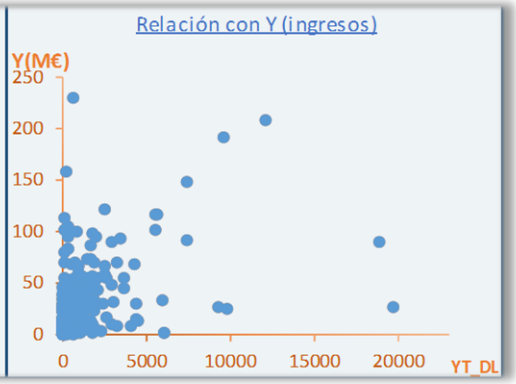
YT_L. Debemos arreglar los Outliers antes de realizar los modelos de estimación.



YT_DL. Debemos arreglar los Outliers antes de realizar los modelos de estimación.

52. YT_DL	
Número de Dislikes que el trailer oficial de la película tiene hasta 10 días antes del estreno	
Origen:	API WA-YouTube
Tipo:	Contínua
Mín -	0
Máx -	19.732
Media	847
Desv.Estándar	1.939
Mediana	240

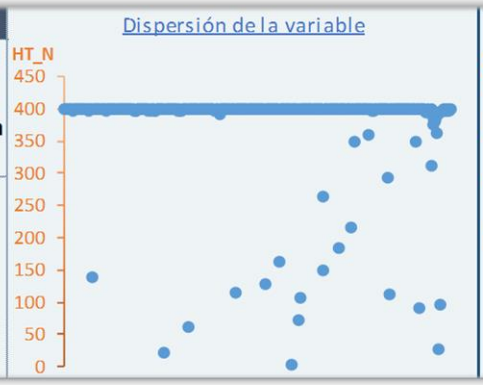




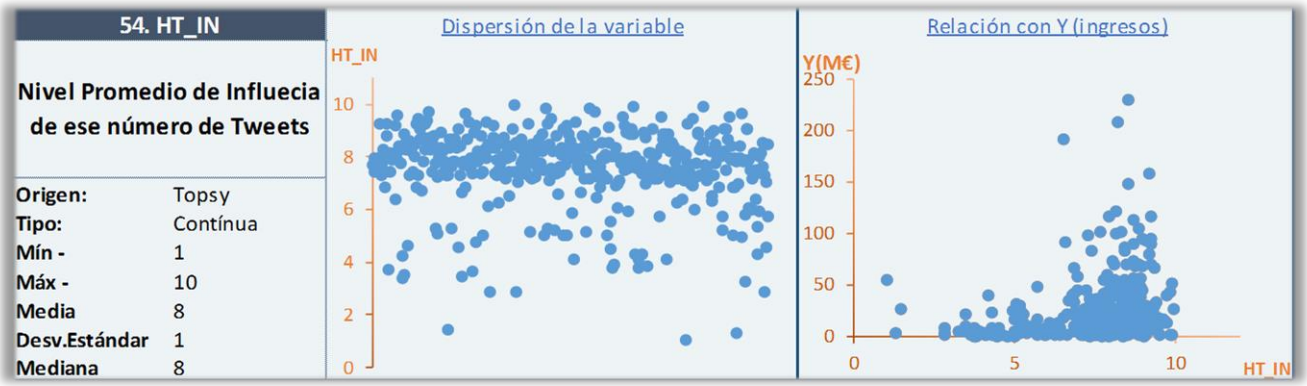
La última información extraída se refiere a términos de búsqueda o Hashtag relacionados con cada película, extraídos de Twitter a través de una cuenta de prueba de Topsy. De cada película intentamos extraer 401 tweets; debido a limitaciones computacionales y sobretodo por la propia cuenta de prueba que apenas nos permite extraer 7000 tweets al día.

En total por esta vía contamos con 162.052 tweets, que sacamos en unos 24 días. El máximo es 401, lo que no significa que hayamos llegado a esa cifra siempre. A veces, películas menos comerciales nos otorgan una cantidad menor. De cada uno de esos Tweets individualmente tenemos información sobre la influencia de quien lo ha escrito en función de un algoritmo desarrollado por Topsy, información sobre citaciones e impresiones que la comunidad ha realizado sobre ese Tweet.

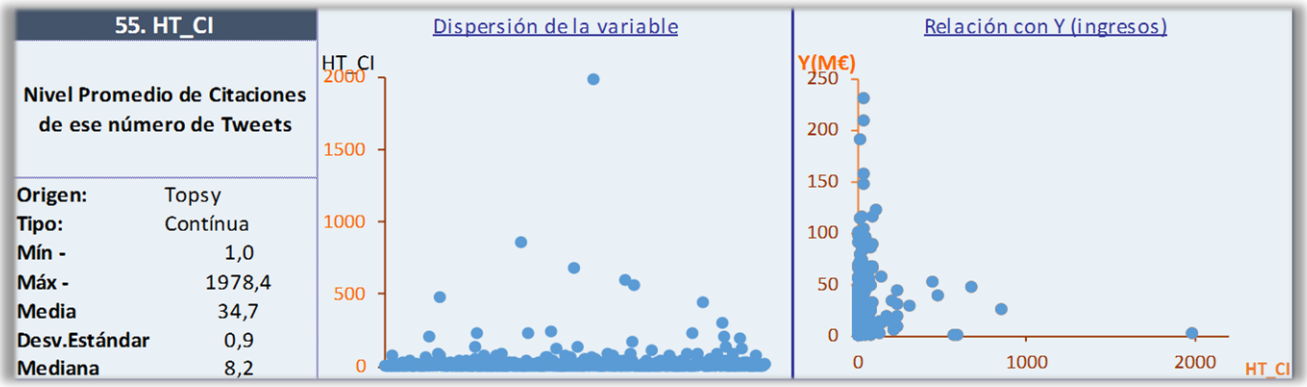
53. HT_N	
Nº de Tweets obtenidos de Topsy del Hastag/term.busqueda de la película entre 30 y 10 días	
Origen:	Topsy
Tipo:	Contínua
Mín -	3
Máx -	401
Media	388
Desv.Estándar	59
Mediana	401



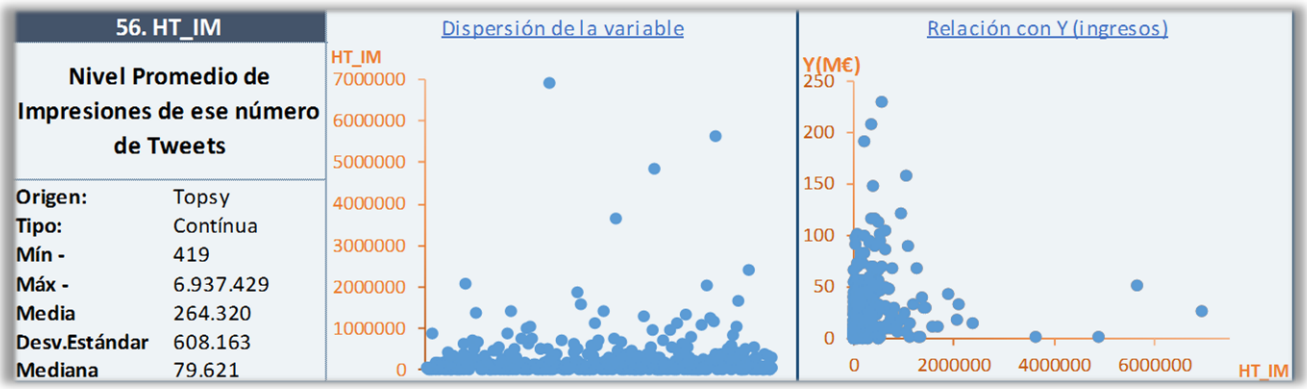
HT_IN. La influencia de los que escriben los tweets está expresada de 0 a 10 en un algoritmo dado por Topsy. Para este estudio sacamos la influencia promedia.



HT_CI. Debemos arreglar el inconveniente con la variable Outliers.

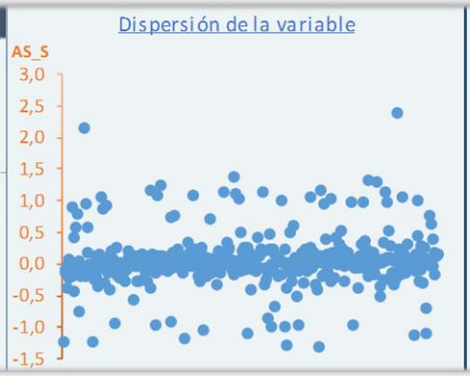


HT_IM. Debemos arreglar los Outliers antes de realizar los modelos de estimación.

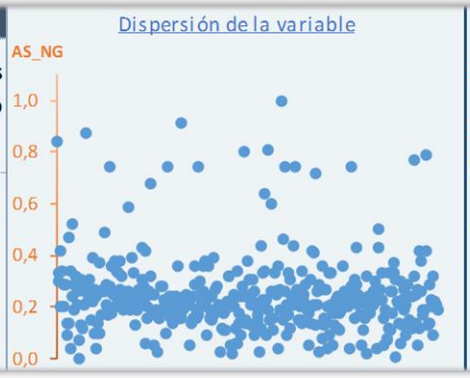


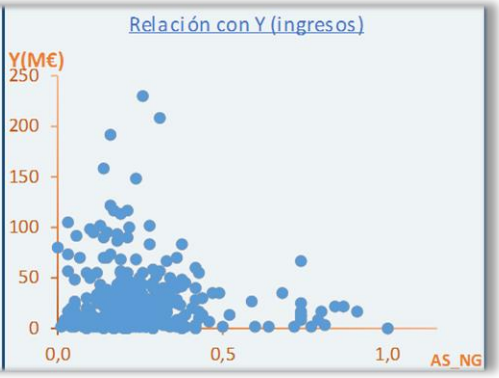
Posteriormente, para extraer una mayor información de los propios textos de los tweets, realizamos un análisis de sentimientos. Y obtenemos información sobre el score promedio de todos los textos para cada película; así como el porcentaje de tweets positivos, negativos y neutros.

57. AS_S	
Score Promedio del Sentimiento de ese número de Tweets	
Origen:	Sentimen.Analysis
Tipo:	Contínua
Mín -	-1,3
Máx -	2,4
Media	0,1
Desv.Estándar	0,4
Mediana	0,0

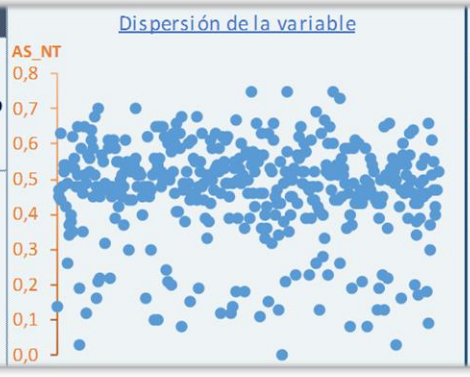


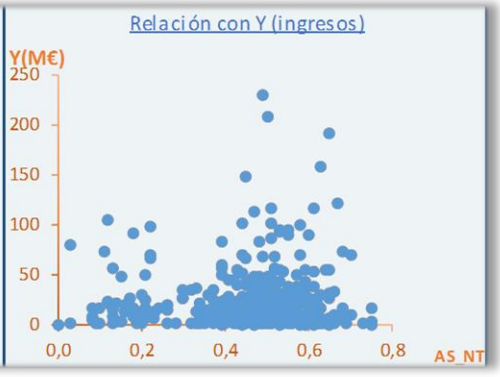
58. AS_NG	
% de Tweets Negativos escritos junto al Hashtag de la película o con ese término de búsqueda	
Origen:	Sentimen.Analysis
Tipo:	Contínua
Mín -	0,0
Máx -	1,0
Media	0,2
Desv.Estándar	0,1
Mediana	0,2



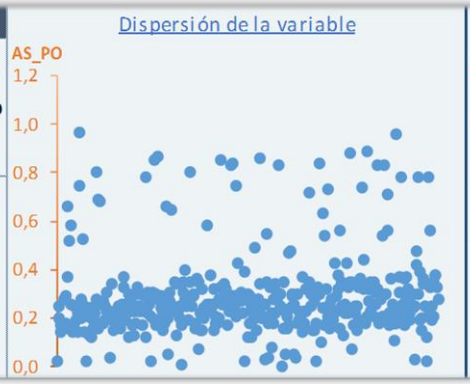


59. AS_NT	
% de Tweets Neutros escritos junto al Hashtag de la película o con ese término de búsqueda	
Origen:	Sentimen.Analysis
Tipo:	Contínua
Mín -	0,0
Máx -	0,8
Media	0,5
Desv.Estándar	0,1
Mediana	0,5





60. AS_PO	
% de Tweets Positivos escritos junto al Hashtag de la película o con ese término de búsqueda	
Origen:	Sentimen.Analysis
Tipo:	Contínua
Mín -	0,0
Máx -	1,0
Media	0,3
Desv.Estándar	0,2
Mediana	0,3



3 MODELOS DE REGRESIÓN LINEAL

El primer modelo de regresión se va a basar en el supuesto que la variable dependiente, se explica de manera lineal a partir de las variables independientes.

En estadística la regresión lineal múltiple o ajuste lineal múltiple es un método matemático que modela la relación entre una variable dependiente Y, las variables independientes Xi y un término aleatorio ε. Este modelo puede ser expresado como:

$$Y_t = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \varepsilon$$

- Yt: variable dependiente o explicada.
- X1, X2, …, Xp : variables independientes o explicativas.
- BETA0, BETA1, BETA2, …,BETAp : parámetros, miden la influencia que las variables independientes tienen sobre la dependiente.

Dónde BETA0 es la intersección o término "constante", las BETA1,2…p son los parámetros respectivos a cada variable independiente, y p es el número de parámetros independientes a tener en cuenta en la regresión. La regresión lineal puede ser contrastada con la regresión no lineal[4].

Vamos a distinguir dos tipos diferentes de creación de modelos: En primer los modelos simples, con los que nos referimos a aquellos en los que las variables entran directamente en el algoritmo, sin realizar ningún tipo de transformación extra sobre ellas, ni buscar relaciones entre las variables. De éstos aplicaremos varias modalidades en función de los algoritmos que entren en los modelos. Y en segundo lugar, crearemos los modelos con iteraciones, que son aquellos con los que intentamos encontrar relaciones ocultas entre las categorías de las variables nominales y las variables continuas.

En este sentido, para la mayoría de los grupos de variables sacamos 12 modelos, que provienen de las posibles elecciones provenientes de escoger entre tres criterios de seleccionar las variables y cuatro criterios para comparar los modelos en función de los parámetros y errores que cada uno genere.Criterios de elegir las variables que formarán parte del modelo:

- StepWise: Esta elección de variable comienza el modelo con 0 variables, y en cada paso decide si eliminar alguna variable/efecto o introducir. Se escoge la acción que mejora más el criterio general. El algoritmo se detiene cuando el criterio no se puede seguir mejorando.

[4] Regresión lineal – Wikipedia: https://es.wikipedia.org/wiki/Regresi%C3%B3n_lineal

- BackWard: Esta forma de elección de variables se refiere a comenzar el modelo con todas las variables, e ir comparando los resultados cada vez que se quita una variable, hasta que dicho test no mejore el resultado según el criterio de comparación de modelos elegido.
- FordWard: Esta forma de elección de variables se refiere a comenzar el modelo sin variables, e ir comparando los resultados cada vez que se añade una nueva variable, hasta que dicho test no mejore el resultado según el criterio de comparación de modelos elegido.

El criterio de comparación de los modelos, se refiere a diferentes formulaciones estadísticas basadas en la población (n), el error (SSE) y número de parámetros incluidos en el algoritmo (P). Se han elegido cuatro pues son aquellos más utilizados y permiten evitar el sobreajuste ocurridos por introducir demasiadas variables (lo que podría ocurrir fácilmente en nuestro modelo, pues contamos con 61 variable y 400 observaciones), y es que que cuantas más variables se introduzcan, crea más penalizaciones. Así, podemos definir cada uno de ellos como:

Tabla 1: Fórmulas de los criterios de selección de variables.

SBC Schwarz's Bayesian Criterion SBC $n\ln\left(\frac{SSE}{n}\right)+p\ln(n)$	**AIC** Criterio de Información de Akaike AIC $n\ln\left(\frac{SSE}{n}\right)+2p$
BIC Criterio de Información Bayesiano BIC $n\ln\left(\frac{SSE}{n}\right)+2(p+2)q-2q^2$ where $q=\frac{n\hat{\sigma}^2}{SSE}$	**SL:** Nivel de Significancia Aproximación tradicional basada en la probabilidad de cometer un error determinado

Se aplicará al modelo, a través del Proc glmselect de SAS, de la siguiente manera:

Código 1: Ejemplo de codificación de la regresión lineal en SAS.

```
proc glmselect data=local.masterdb;
  class EST MON X3D RAT L_E L_O C_AO C_EU C_RA C_USA G_AC G_AN G_B G_C G_D
        G_F G_H G_M G_R G_T WA_UO WA_CO WA_MM CT_T YT_T ;
  model Y= AS_NG AS_NT AS_PO AS_S BUDG CT_F CT_N CT_R C_N DAY G_N HT_CI HT_IM
           HT_IN HT_N LET L_N OC RUNT WA_CC WA_CR WA_CV WA_N WA_UC WA_UR WA_UV
           YT_DL YT_L YT_S YT_V EST MON X3D RAT L_E L_O C_AO C_EU C_RA C_USA YEAR YT_D
           G_AC G_AN G_B G_C G_D G_F G_H G_M G_R G_T WA_UO WA_CO WA_MM CT_T YT_T
  /selection=stepwise(select=SBC choose=SBC);
  run;
```

Antes de comenzar con la explicación de los modelos, explicaremos cómo compararemos los diferentes algoritmos creados, y que nos permitirá ponderar uno de ellos como el mejor. Para ello aplicaremos la validación cruzada. La validación cruzada o cross-validation es una técnica utilizada para evaluar los resultados de un análisis estadístico y garantizar que son independientes de la partición entre datos de entrenamiento y prueba. Consiste en repetir y calcular la media aritmética obtenida de las medidas de evaluación sobre diferentes particiones. Se utiliza en entornos donde el objetivo principal es la predicción y se quiere estimar cómo de preciso es un modelo que se

llevará a cabo a la práctica. Es una técnica muy utilizada en proyectos de inteligencia artificial para validar modelos generados. (“Validación cruzada - Wikipedia, la enciclopedia libre,” n.d.)

3.1 MODELOS COMPLETOS CON VARIABLE DEPENDIENTE NORMAL

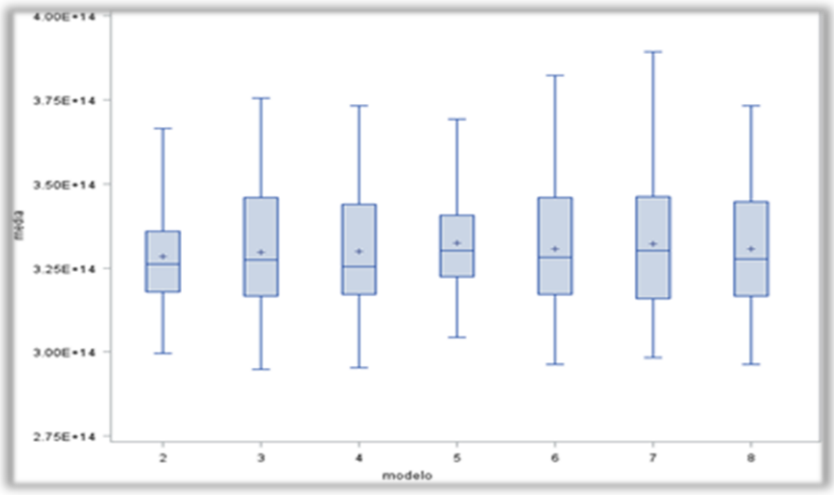
En primer lugar incorporamos las 61 variables de las que disponemos para intentar hallar un modelo de regresión óptimo en la estimación del ingreso de una película en su primer fin de semana. Aplicando los doce modelos especificados anteriormente, logramos el siguiente cuadro.

Tabla 2: Modelos R. Lineal para la variable dependiente normal con todas las variables independientes.

MODELO	CRIT	MODELO	MEJOR MODELO
STEPWISE	SBC	final2	BUDG HT_IM OC WA_CR WA_N WA_UV YT_L YT_S WA_UO WA_CO
	AIC	final3	AS_NT BUDG HT_IM OC WA_CC WA_CR WA_N WA_UC WA_UV YT_D YT_L YT_S L_E C_USA G_H G_T WA_UO WA_CO WA_MM YT_T
	BIC	final4	AS_NT BUDG HT_IM OC WA_CR WA_N WA_UC WA_UV YT_L YT_S C_USA G_T WA_UO WA_CO WA_MM
	SL	final5	BUDG HT_IM OC WA_CR WA_N WA_UV YT_L YT_S WA_CO
FORWARD	SBC	final2	BUDG HT_IM OC WA_CR WA_N WA_UV YT_L YT_S WA_UO WA_CO
	AIC	final6	AS_NT BUDG HT_CI HT_IM OC WA_CC WA_CR WA_N WA_UC WA_UV YT_D YT_L YT_S L_E C_USA G_H G_T WA_UO WA_CO WA_MM YT_T
	BIC	final4	AS_NT BUDG HT_IM OC WA_CR WA_N WA_UC WA_UV YT_L YT_S C_USA G_T WA_UO WA_CO WA_MM
	SL	final5	BUDG HT_IM OC WA_CR WA_N WA_UV YT_L YT_S WA_CO
BACKWARD	SBC	final2	BUDG HT_IM OC WA_CR WA_N WA_UV YT_L YT_S WA_UO WA_CO
	AIC	final7	BUDG HT_IM LET L_N OC WA_CC WA_CR WA_N WA_UC WA_UV YT_D YT_L YT_S L_O C_AO C_EU G_H WA_UO WA_CO WA_MM YT_T
	BIC	final8	BUDG HT_IM OC WA_CR WA_N WA_UC WA_UV YT_L YT_S C_EU WA_UO WA_CO WA_MM
	SL	final5	BUDG HT_IM OC WA_CR WA_N WA_UV YT_L YT_S WA_CO

Como se puede observar, algunos de los modelos se repiten, sobretodo por la fuerza de los criterios SBC y SL. Esto significa que esos modelos han sido escogidos como el mejor algoritmo posible por diferentes caminos, lo que a priori podría mostrarse como que son mejores modelos. Como ya se ha indicado, para escoger el mejor modelo vamos a realizar Validación Cruzada. Así, llegamos a una solución que nos demuestra que todos los modelos son parecidos en cuanto a la media del error, por ello vamos a escoger aquel que tenga una menor varianza, siendo el modelo dos el que mejor cumple esta cualidad.

Imagen 3: Error promedio R. Lineal Total



3.2 MODELOS CON SELECCIÓN DE VARIABLES LÓGICAS

Además, después del análisis y exploración de variables que se realizó para depurar la base de datos; llegamos a ciertas conclusiones en el entorno de las relaciones directas de cada variable independiente con la variable dependiente. Por lo que decidimos probar cómo funcionarían los modelos dejando exclusivamente éstas variables.

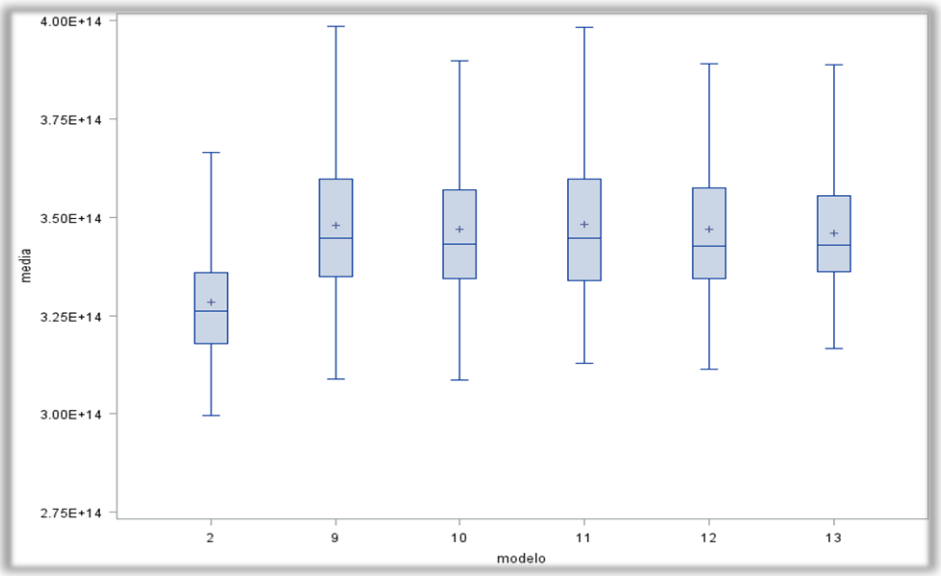
Así, realizamos una elección amplia de variables, quedándonos únicamente con 43, y realizamos los doce modelos explicados anteriormente; llegando a la siguiente tabla de modelos.

Tabla 3: Modelos R.Lineal para la variable dependiente normal con selección lógica de variables.

MODELO	CRIT	MODELO	MEJOR MODELO
STEPWISE	SBC	final9	BUDG HT_IM OC WA_CR WA_UV YT_L WA_UO WA_CO
	AIC	final10	AS_NT BUDG HT_CI HT_IM OC WA_CR WA_UC WA_UV YT_L G_T WA_UO WA_CO WA_MM
	BIC	final11	AS_NT BUDG HT_IM OC WA_CR WA_UC WA_UV YT_L WA_UO WA_CO WA_MM
	SL	final9	BUDG HT_IM OC WA_CR WA_UV YT_L WA_UO WA_CO
FORWARD	SBC	final9	BUDG HT_IM OC WA_CR WA_UV YT_L WA_UO WA_CO
	AIC	final10	AS_NT BUDG HT_CI HT_IM OC WA_CR WA_UC WA_UV YT_L G_T WA_UO WA_CO WA_MM
	BIC	final11	AS_NT BUDG HT_IM OC WA_CR WA_UC WA_UV YT_L WA_UO WA_CO WA_MM
	SL	final9	BUDG HT_IM OC WA_CR WA_UV YT_L WA_UO WA_CO
BACKWARD	SBC	final9	BUDG HT_IM OC WA_CR WA_UV YT_L WA_UO WA_CO
	AIC	final12	AS_NG AS_S BUDG HT_CI HT_IM OC WA_CR WA_UC WA_UV YT_L G_T WA_UO WA_CO WA_MM
	BIC	final13	AS_NG AS_S BUDG HT_IM OC WA_CR WA_UC WA_UV YT_L WA_UO WA_CO WA_MM
	SL	final9	BUDG HT_IM OC WA_CR WA_UV YT_L WA_UO WA_CO

Pasamos pues a comparar estos modelos nuevos, con el número 2 que indicamos en la tabla anterior como el mejor modelo; para comprobar que en ningún caso nos llevan a una mejora en la media del error, y por tanto descartamos éstas opciones.

Imagen 4: Error promedio R. Lineal lógico



3.3 MODELOS CON SELECCIÓN DE VARIABLES LÓGICAS REDUCIDO

Una vez, sabiendo que tenemos demasiadas variables en relación al número de observaciones, realizamos una selección de variables de la misma forma que en punto anterior, pero siendo aún más estrictos, para aplicar un número más reducido de variables. En este caso trabajamos únicamente con 35 de ellas.

De nuevo aplicamos los doce modelos posibles, obteniendo la siguiente tabla de algoritmos de abajo. Como se puede observar, el número de variables cada vez es menor; lo que es algo positivo al realizar modelos más simples, y que por definición serán más consistentes y robustos.

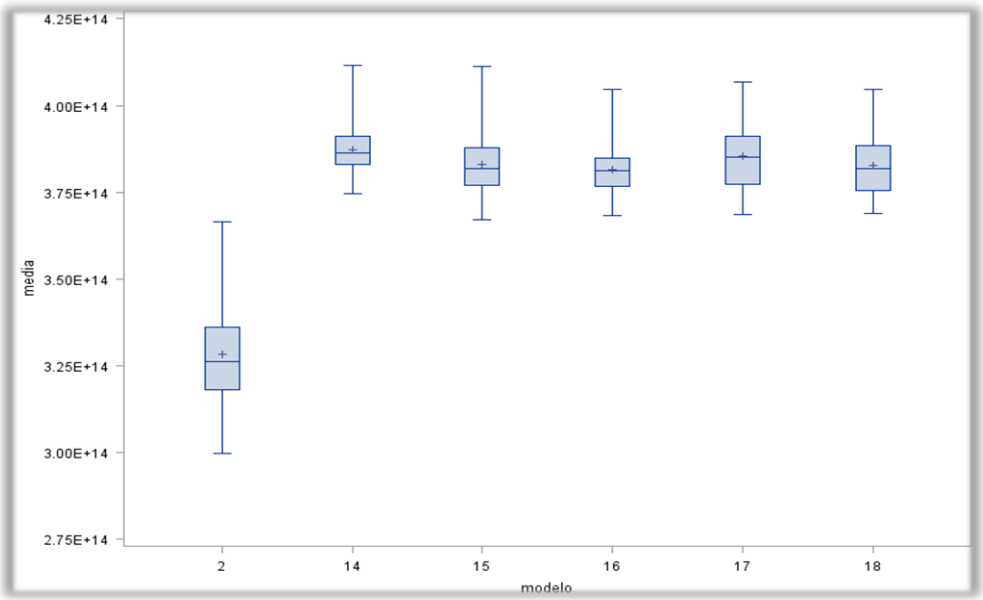
Tabla 4: Modelos R. Lineal para la variable dependiente normal con selección lógica reducida de variables.

MODELO	CRIT	MODELO	MEJOR MODELO
STEPWISE	SBC	final14	BUDG HT_IM OC YT_L
	AIC	final15	AS_NT BUDG HT_CI HT_IM OC RUNT YT_L
	BIC	final15	AS_NT BUDG HT_CI HT_IM OC RUNT YT_L
	SL	final14	BUDG HT_IM OC YT_L
FORWARD	SBC	final14	BUDG HT_IM OC YT_L
	AIC	final15	AS_NT BUDG HT_CI HT_IM OC RUNT YT_L
	BIC	final15	AS_NT BUDG HT_CI HT_IM OC RUNT YT_L
	SL	final14	BUDG HT_IM OC YT_L
BACKWARD	SBC	final16	AS_NG AS_NT AS_PO BUDG HT_IM OC YT_L
	AIC	final17	AS_NG AS_NT AS_PO AS_S BUDG HT_IM OC RUNT YT_DL YT_L YT_V
	BIC	final18	AS_NG AS_NT AS_PO AS_S BUDG HT_IM OC RUNT YT_L
	SL	final16	AS_NG AS_NT AS_PO BUDG HT_IM OC YT_L

Y de nuevo comparamos estos nuevos modelos, con el número 2 que hasta el momento se ha mostrado como nuestro mejor modelo; llegando a la siguiente conclusión, la preselección de

variables no ayuda en la elección de un mejor modelo. Si bien si crea modelos más simples con menos variables, lo cual es bueno; el hecho de que genere una media del error tan negativa; nos hará decantarnos por el modelo 2 de nuevo como nuestro mejor modelo.

Imagen 5: Error promedio R. Lineal lógico reducido



3.4 MODELO CON SELECCIÓN DE VARIABLES DE REDES SOCIALES

Dada la importancia de las redes sociales en nuestro análisis, queremos probar cómo funcionaría un modelo aplicando únicamente aquellas generadas a través de YouTube, Twitter, y el análisis de sentimientos asociado a los Tweets. Los modelos son:

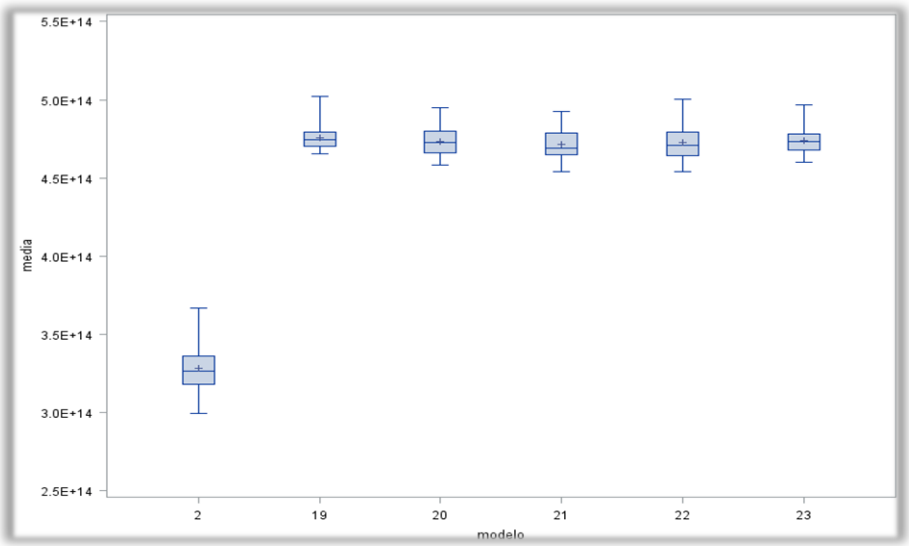
Tabla 5: Modelos R. Lineal para la variable dependiente normal con selección redes sociales.

MODELO	CRIT	MODELO	MEJOR MODELO
STEPWISE	SBC	final19	HT_IM HT_IN YT_V
	AIC	final20	HT_IM HT_IN YT_DL YT_L YT_V
	BIC	final20	HT_IM HT_IN YT_DL YT_L YT_V
	SL	final19	HT_IM HT_IN YT_V
FORWARD	SBC	final19	HT_IM HT_IN YT_V
	AIC	final20	HT_IM HT_IN YT_DL YT_L YT_V
	BIC	final20	HT_IM HT_IN YT_DL YT_L YT_V
	SL	final19	HT_IM HT_IN YT_V
BACKWARD	SBC	final21	AS_NG AS_NT AS_PO HT_IM HT_IN YT_L YT_V
	AIC	final22	AS_NG AS_NT AS_PO AS_S HT_IM HT_IN YT_DL YT_L YT_V
	BIC	final22	AS_NG AS_NT AS_PO AS_S HT_IM HT_IN YT_DL YT_L YT_V
	SL	final23	AS_NG AS_NT AS_PO HT_IM HT_IN YT_V

Como se observa en la gráfica siguiente, las variables de redes sociales únicamente no nos mejoran el error promedio logrado en la estimación de la variable independiente. Esto solo implica que las

variables escogidas de redes sociales, no nos ayudan a estimar en formato lineal, deberemos por tanto probar como funciona a nivel de redes. Y de nuevo, nos quedamos con el modelo inicial simple número 2, como nuestro mejor modelo a nivel de estimación del error.

Imagen 6: Error promedio R. Lineal variables de redes sociales



3.5 MODELO CON VARIABLE DEPENDIENTE LOGARÍTMICA

A menudo son necesarias transformaciones sobre variables continuas para conseguir linealidad. Dada la amplitud de nuestra variable independiente, vamos a realizar una transformación de la variable dependiente como logaritmo, para ver si logramos una mejor aplicabilidad de las variables independientes.

Al transformarla, los parámetros no tienen el mismo significado. Es necesario deshacer la transformación después de plantear el modelo para obtener un *SSE* real.

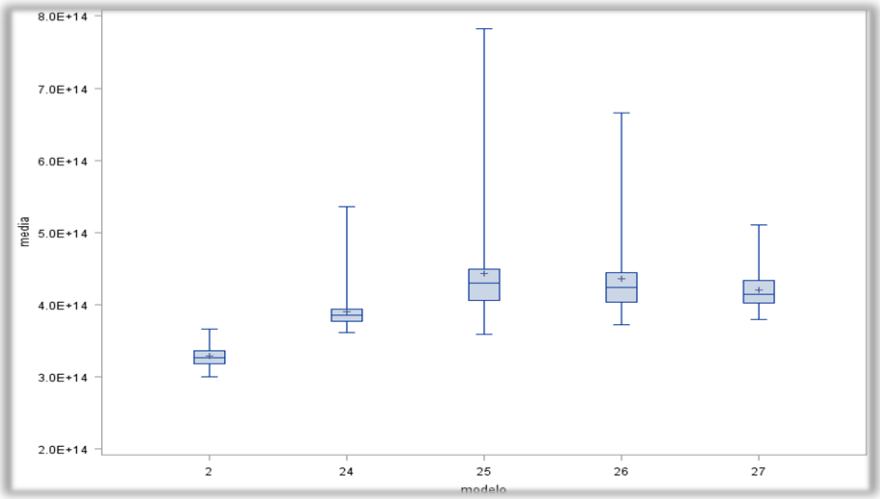
Dado que hasta ahora *StepWise* ha sido el modelo de selección de variables que mejor se ha portado al nivel de la validación cruzada, para este punto solo vamos a comparar los datos sacados de este modelo.

Tabla 6: Modelos R. Lineal para la variable dependiente logarítmica con total de variables.

MODELO	CRIT	MODELO	MEJOR MODELO
STEPWISE	SBC	final24	HT_IN OC WA_CV WA_N YT_S YT_V
	AIC	final25	AS_NT CT_N HT_IM HT_N LET OC WA_CV WA_N WA_UV YEAR YT_D YT_DL YT_L YT_S YT_V EST C_USA G_M
	BIC	final26	AS_NT CT_N HT_IM HT_N OC WA_CV WA_N WA_UV YEAR YT_DL YT_L YT_S EST
	SL	final27	HT_IM HT_N OC WA_CV WA_N YT_L YT_S EST

No siempre la liberalización a través de la transformación de la variable dependiente consigue una mejor predicción en términos de SSE (incluso sobre datos test). En este caso, trabajar con el logaritmo solo nos lleva a tener una menor robustez en nuestros modelos, pues la varianza se amplia. Y la media del error también empeora. Por tanto, nuestro modelo inicial 2 sigue siendo el mejor hasta este punto.

Imagen 7: Error promedio R. Lineal variables dependiente logarítmica.



3.6 MODELOS CON ITERACIONES

La realización de modelos con iteraciones nos va a permitir sacar mayor utilidad a las variables nominales, pues se trata de unir las categorías de cada una de las variables nominales, con toda una de las variables continuas; es decir, unir una a una cada categoría nominal con cada una de las variables continuas. Con ello multiplicamos el número de variables que podemos crear desde las 61 simples, hasta 1.157 variables. Con apenas 401 observaciones, es mejor realizar una preselección, y escoger únicamente aquellas que parecen que a priori pudieran mostrar mejor predictibilidad. Así, realizamos un filtro a través del *FValue*, para trabajar únicamente con las más representativas. Este valor *FValue* se calcula a partir de la distribución *Fisher-Snedecor*, y representa la comprobación de la igualdad de varianzas de dos poblaciones normales; técnica que permite detectar la existencia o inexistencia de diferencias significativas entre muestras diferentes y que es, por tanto, esencial, en todos aquellos casos en los que se quiere investigar la relevancia de un factor en el desarrollo y naturaleza de una característica. (“DISTRIBUCIÓN F DE SNEDECOR,” n.d.)

Una vez realizado ese análisis, se decide escoger tres grupos de variables. Todos ellos aplicados bajo el criterio *StepWise-SBC* y con la variable dependiente normal; pues se ha mostrado como la más robusta.

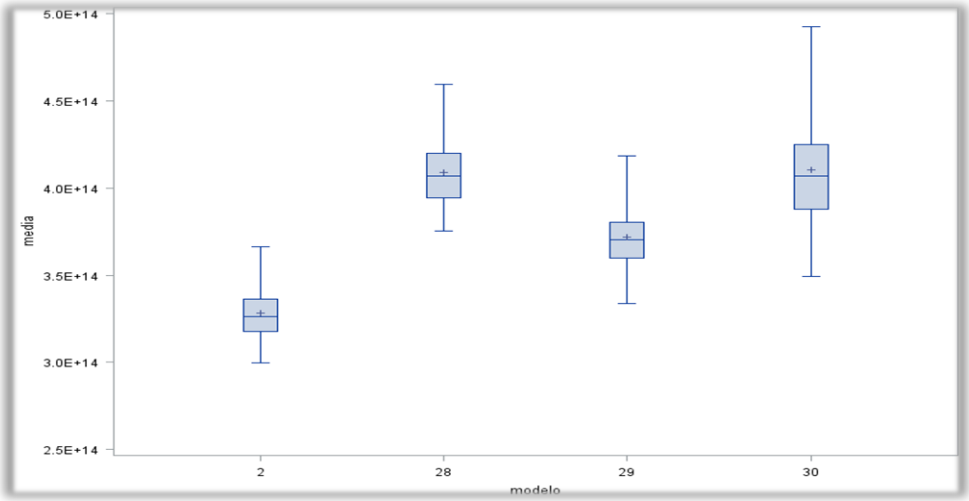
El primer modelo contiene las primeras 50 variables, y tienen un Fvalue mayor a 64 puntos. El segundo modelo, tiene 100 variables con un FValue mayor a 44 puntos. Y, por último, el tercer modelo que realizamos tiene 227 variables, con un FValue mayor a 20 puntos.

Tabla 7: Modelos R. Lineal para la variable dependiente normal con iteraciones de variables

MODELO	CRIT	MODELO	MEJOR MODELO
STEPWISE SBC	50	final24	OC*WA_CO BUDG*G_F BUDG*C_USA WA_UV YT_V*G_B YT_V*G_AN YT_V*L_O
	100	final25	OC BUDG*C_RA YT_V*X3D YT_V*G_B YT_V*G_F BUDG*YT_T HT_IM*G_F WA_UV*C_AO
	227	final26	BUDG*G_B BUDG*C_AO YT_L*G_AN OC*C_USA HT_IM*G_F WA_UV*C_USA WA_UV*C_AO WA_UV*G_B WA_UV*WA_MM YT_S WA_CR WA_N

Como se observa en la gráfica siguiente, tampoco de esta manera logramos mejorar el primero modelo simple realizado con todas las variables y Stepwise-SBC.

Imagen 8: Error promedio R. Lineal variables con iteraciones.



3.7 SELECCIÓN DEL MEJOR MODELO LINEAL

En definitiva, tras analizar múltiples opciones para crear un modelo que nos ayude a estimar mejor los ingresos de una película en su primer fin de semana de estreno, podemos concluir que el mejor algoritmo es aquel logrado a través del modelaje *StepWise*, bajo el *Schwarz criterion (SBC);* incluyendo todas las variables disponibles y permitiendo a SAS que escoja las más relevantes.

De esta forma, el modelo estaría formado por 10 variables, que se descompone en (A) dos variables clásicas (BUDG, OC); (B) cinco variables de representación de las votaciones en IMDB (WA_CR, WA_N, WA_UV, WA_UO, WA_CO); (C) dos variables de YouTube (YT_L, YT_S) y (D)una variable de Twitter, en específico de la información conseguida de su *hashtag* (HT_IM).

Además, podemos decir que con este modelo logramos reducir el error hasta tener una varianza de 3,25x10^14. La desviación estándar o desviación típica es la raíz cuadrada de la varianza. Es decir, la raíz cuadrada de la media de los cuadrados de las puntuaciones de desviación.

4 MACHINE LEARNING: REGRESIÓN CON ALGORITMOS DE REDES NEURONALES

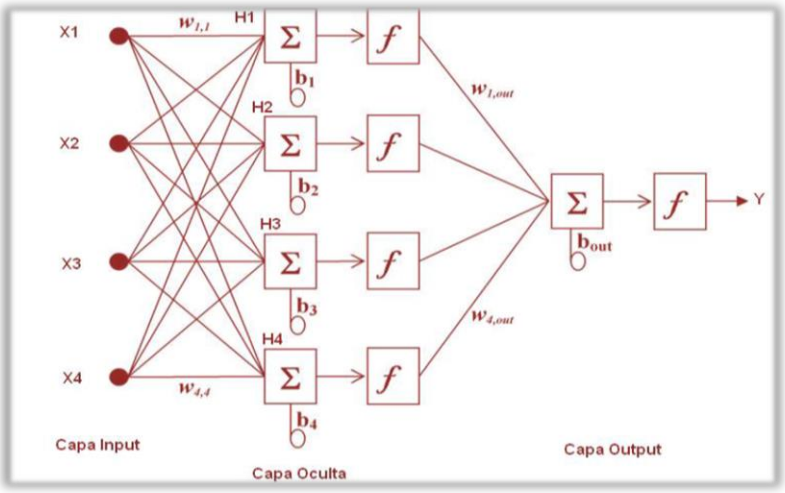
Dada nuestra variable dependiente, la regresión lineal presenta un problema de heterocedasticidad[5].

La heterocedasticidad implica que el modelo lineal presenta una varianza de las perturbaciones que no es constante a lo largo del tiempo; es decir, se presentan diferentes tendencias según la observación por el tipo de variable que estamos tratando debido a que las películas no presentan un comportamiento homogéneo. De ella se deriva que los datos con los que se trabaja son heterogéneos, ya que provienen de distribuciones de probabilidad con distinta varianza. La principal consecuencia de esta heterocedasticidad, es el incumplimiento de una de las hipótesis básicas sobre las que se asienta el modelo de regresión lineal.

Para solucionar el problema de la heterocedasticidad, aplicamos las redes neuronales. Las redes neuronales son un paradigma de aprendizaje y procesamiento automático, que se encarga de entrenar una red mediante la creación de una serie de nodos ocultos; y funciones internas entre esos nodos, inputs (variables incluidas por nosotros) y output (función final de aprendizaje para estimar). Los métodos utilizados son técnicas de optimización numérica, que van variando los valores de los parámetros de manera iterativa, hasta cumplir el objetivo de optimización, que en nuestro caso será minimizar la función de error.

Imagen 9: Representación gráfica del modelo de Redes Neuronales

[5] Heterocedasticidad – Wikipedia: https://es.wikipedia.org/wiki/Heterocedasticidad



Como ya se ha indicado, la utilización de SAS 9.4 nos va a permitir ajustar más parámetros de los que R es capaz; lo que a priori nos permitirá llegar a una mejor solución.

Antes de desarrollar cada uno de los parámetros, es necesario indicar una nueva limitación con la que contamos, y es el limitado número de observaciones propio del sector en el que estamos trabajando (no se estrenan tantas películas relevantes a lo largo de un año; y ampliar el límite temporal podría llevarnos a modificar el criterio de decisión puesto que las redes sociales no eran tan importantes). Vamos a seguir el manual SAS, que nos dice que debemos contar con entre 5 y 25 observaciones por categoría; así seguiremos la función de creación de parámetros de redes; y despejaremos para saber con hasta cuantos nodos podemos trabajar; llegando al siguiente cuadro:

Tabla 8: Tabla teórica de nodos promedio y máximo a crear según número de variables.

MODELO	Nº Observ.	Nº Var	Nº Nodos Medio	Nº Nodos Máx
M3	401	15	2	5
M4	401	10	2	7
M5	401	12	2	6
M6	401	13	2	5

FÓRMULA: h(k+1)+h+1

k=número de nodos input (variables x)

h= número de nodos ocultos

Así, teniendo en cuenta un promedio de 15 observaciones por parámetros, únicamente podríamos incluir 2 nodos. Si llevamos el modelo al límite, y permitimos únicamente 5 observaciones por parámetro, el número de nodos se incrementaría hasta alcanzar entre 5 y 7, en función del número de variables.

Teniendo esto en cuenta, los parámetros con los que vamos a ajustar la red neuronal son:

- **Set de Variables**: Contamos con 61 variables; para evitar sobreajustes vamos a trabajar únicamente con aquellas más influentes sobre la variable dependiente según los resultados

de la regresión lineal. Así se crean 4 modelos de variables con entre 10 y 15 variables (ver punto 4.1).

- **Número de semillas**: Este parámetro se refiere al número de veces que se repite el análisis de validación cruzada, modificando los grupos de observaciones train/test, para conseguir unos resultados más robustos. Debido a limitaciones computacionales, vamos a trabajar únicamente con 15 semillas, variando la misma desde 12345 hasta 12360, mediante una macro que automatice el análisis a través de un bucle.

Código 2: Ejemplo de codificación de la semilla en Redes Neuronales.

```
,sinicio=12345
,sfinal=12360
```

- **Número de Capas**: De nuevo, nuestra limitación en el número de observaciones nos hará trabajar únicamente con una capa oculta de nodos. Como es la forma que tiene SAS de trabajar por defecto, no se le añade nada al código.
- **Número de Nodos**: Dada nuestra limitación en el número de observaciones, buscaremos modelos con el menor número de nodos posibles para evitar sobreajuste en la medida de lo posible. A pesar de que teóricamente no debemos trabajar con más de 7 nodos, nuestras pruebas irán desde 1 hasta 15 nodos. Esta parte del trabajo se realizará mediante una macro, que cree un bucle que automatice el trabajo.

Código 3: Ejemplo de codificación del número de nodos en Redes Neuronales.

```
%macro nodosvalcruza(ini=,fin=,increme=);
%do nod=&ini %to &fin %by &increme;

%nodosvalcruza(ini=1,fin=15,increme=1);
```

- **Algoritmo de optimización del aprendizaje:** Trabajaremos con las más importantes, como son: Backpropagation (Bprop), Levenberg-Marquardt (Levmar, es el que trae SAS por defecto), Quasi-Newton (Quanew) y Trust región (Trureg). Para controlar el esfuerzo computacional de las máquinas y evitar lentitud en el proceso debido a la utilización de la memoria caché, esta variable se irá modificando a mano.

Código 4: Ejemplo de codificación del algoritmo de optimización en Redes Neuronales.

```
,meto=bprop mom=0.2 learn=0.1);
,meto= Levmar);
,meto= Quanew);
,meto= Trureg);
```

- **Función de activación:** La función de activación es aquella que relaciona las variables iniciales con los nodos creados; es decir, calcula la activación de la unidad en función de la entrada total y la activación previa. En ocasiones es posible que alguna de ellas no tengan resultado debido a la naturaleza de ciertos datos.

Código 5: Ejemplo de codificación de la función de activación en Redes Neuronales.

```
%macro activalcruza;
%let lista='TANH LOG ARC SIN SOF GAU';
%let nume=6;
%do  i=1 %to &nume;
```

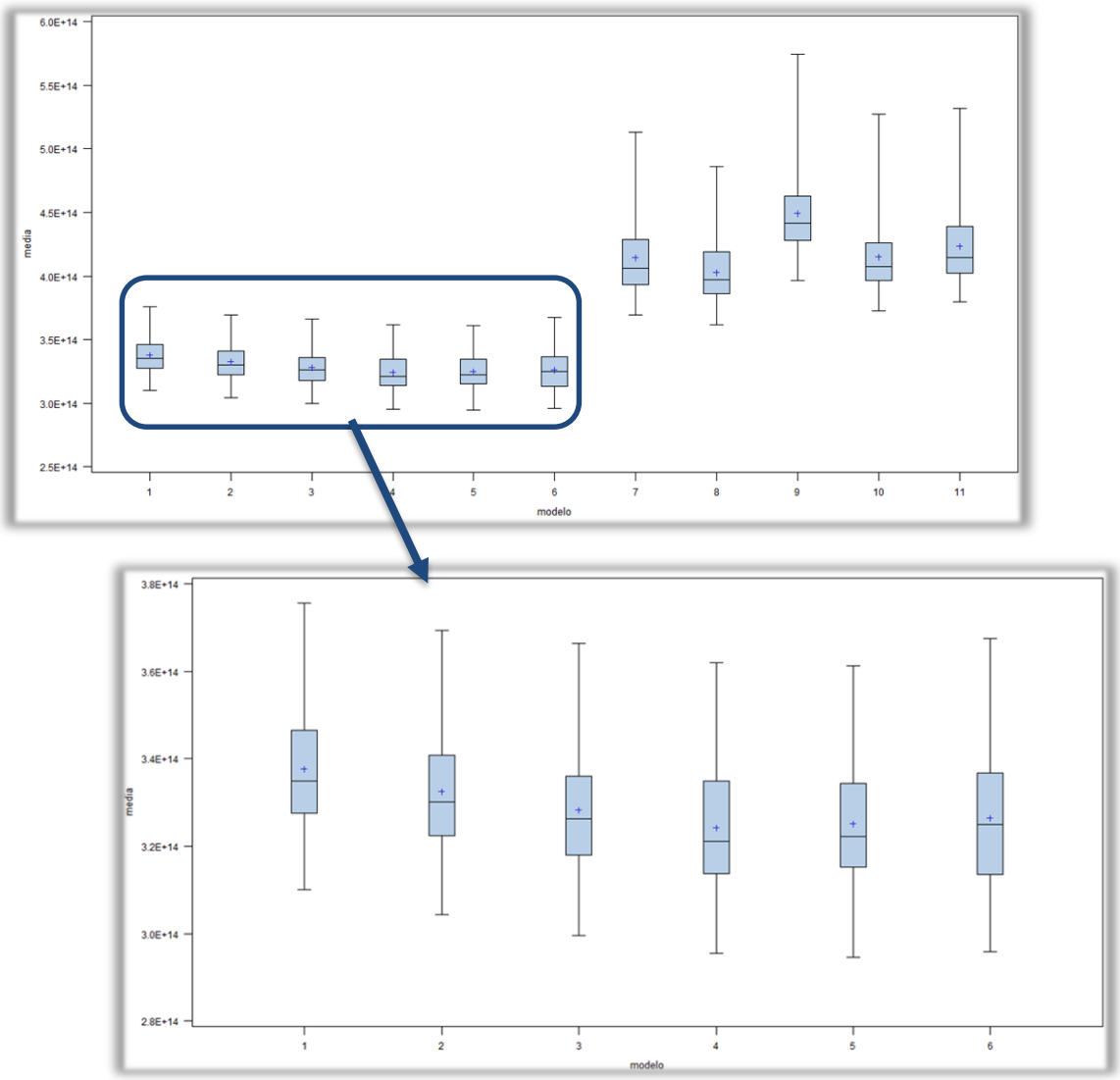
4.1 SELECCIÓN DE VARIABLES.

Como hemos indicado anteriormente, para evitar sobreajuste y optimizar la información que tenemos al trabajar con redes, vamos a realizar una preselección de variables y trabajar únicamente con aquellas que realmente muestran de alguna manera relación en la creación de la variable dependiente.

Para seleccionar las variables realizamos un análisis de selección aleatorio de modelos. Para ello, se realiza un reparto de las observaciones como train-test de 101 maneras diferentes; posteriormente crea con cada uno de ellos el mejor modelo posible para estimar la variable dependiente; finalmente realiza un conteo de cuantas veces se repite cada modelo en esas 101 pruebas, para que escojamos aquellos más repetidos.

Esto lo repetimos, al igual que hemos explicado anteriormente en regresión lineal, para que la elección de modelos y variables sigua diferentes criterios StepWise, Fordward y Backward; así como los criterios SBC, AIC y BIC. Y de igual modo para estimar la variable dependiente normal y la obtenida como logaritmo. De todos los modelos creados, escogemos los 10 mejores, junto al mejor modelo obtenido en regresión lineal. Y realizamos validación cruzada para poder compararlos.

Imagen 10: Elección de los set de variables para Redes Neuronales



De las gráficas anteriores se pueden reconocer cuatro modelos (3, 4, 5, 6) que funcionan levemente mejor que el resto. A partir de ahora realizaremos cuatro análisis de redes paralelos en los que estudiaremos cada uno de los parámetros principales que presentan en el entrenamiento de la información para generar modelos que predigan los ingresos del primer fin de semana, de la manera más acertada posible.

4.2 PROCESO DE TRABAJO

El análisis llevará un proceso similar con todos los grupos de variables, que pasamos a desarrollar a continuación para no repetir en lo sucesivo cada uno de los pasos dados:

En primer lugar, se parte del grupo de variables y se analiza, para cada uno de los algoritmos de optimización, el número óptimo de nodos. Para ello utilizamos una macro que se encargará de automatizar este proceso, obteniendo el error promedio de los nodos de 1 al 15, de 3 en 3.

En segundo lugar, y una vez que conocemos el resultado con menor error promedio, se vuelve a lanzar la macro, esta vez con variaciones de 1 en 1, en torno a ese número de nodos. Nótese en las siguientes gráficas que en número de opciones es diferente para perfeccionar la salida en cuanto a la amplitud del eje; siendo menor el número de nodos si nos encontramos con valores muy dispersos que nos hagan perder información sobre la verdadera bondad del mejor modelo. Una vez conocido el número óptimo de nodos para cada uno de los algoritmos de optimización descritos anteriormente, el tercer paso consiste en mejorar el error del modelo a través de la función de activación. Para ello probamos únicamente el mejor modelo obtenido hasta la fecha; salvo que hubiera varios con resultados similares, en cuyo caso se probarán todos ellos.

El cuarto paso será almacenar ese mejor modelo, para su posterior comparación con los restantes mejores modelos del resto de grupos de variables.

Finalmente, decidimos cual es el mejor modelo de los 4 sub-óptimos. Esto lo realizaremos a través de validación cruzada, y basándonos como veremos en dos criterios, menor promedio del error y menor varianza del error.

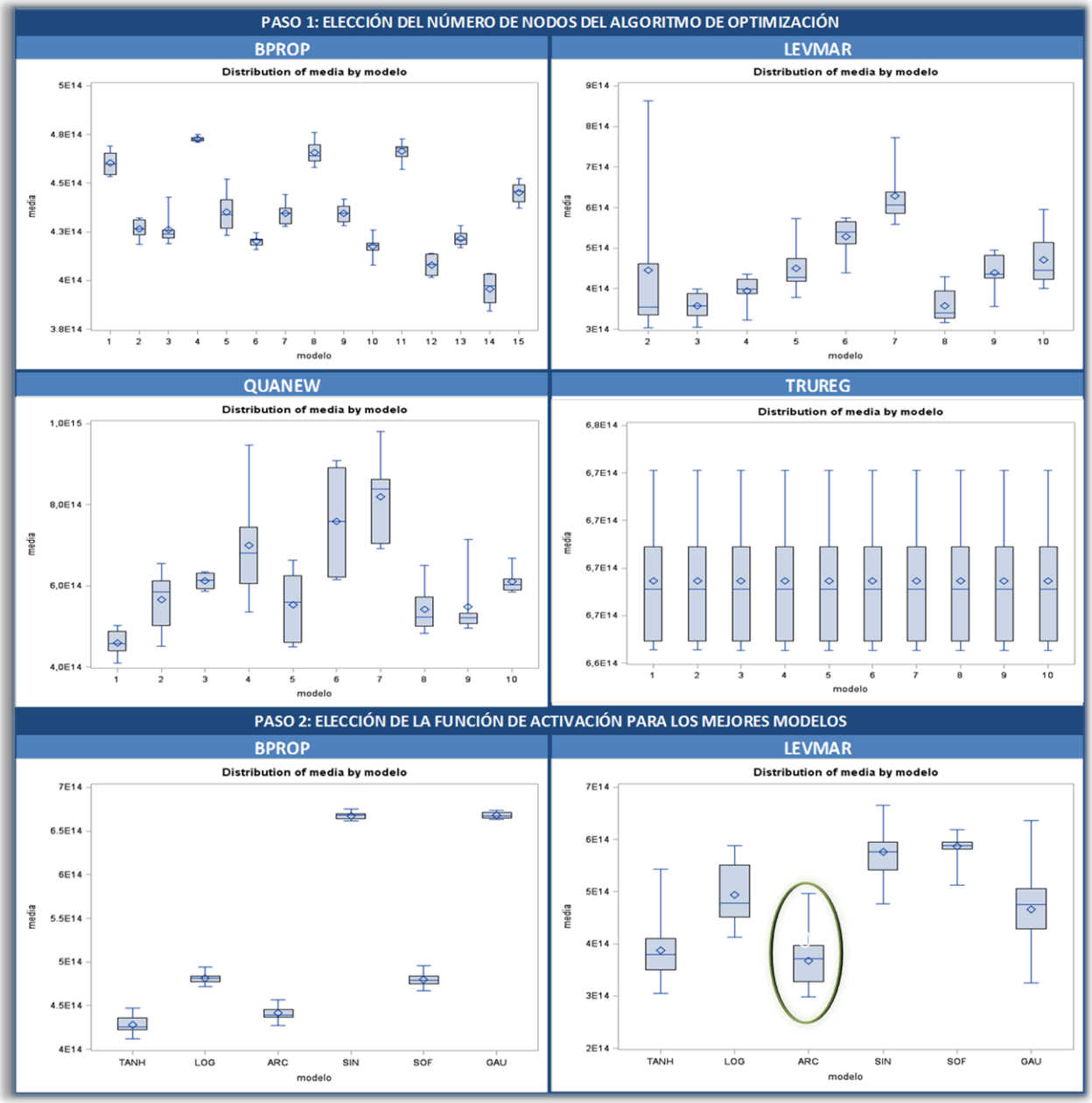
4.3 ENSAYO CON MODELO DE 15 VARIABLES.

Comenzamos con el MODELO 4, que parece el mejor conjunto de variables (y que ya se presentó como nuestro mejor modelo en el análisis de regresión lineal anterior); para realizar el análisis más completo. Este grupo incluye 15 variables, siendo nueve continuas (AS_NT, BUDG, HT_IM, OC, WA_CR, WA_N, WA_UV, YT_L, YT_S) y seis de clase (L_E, C_EU, G_T, WA_UO, WA_CO, WA_MM).

Indicar que, en este comienzo, llegamos a probar el algoritmo de optimización BPROP, con hasta 40 nodos, de 5 en 5. Lo que observamos es que sólo con un máximo de 15 nodos el resultado se parece a los obtenidos por la regresión lineal, por tanto, nos centramos en sacar entre 1 y 15 nodos de 3 en 3. En ese momento notamos que son alrededor de 12 nodos el óptimo; por lo que hacemos zoom de nuevo para obtener el mejor algoritmo con este grupo de variables y BPROP. Tanto la variabilidad como el error promedio nos hace decantarnos por nodos entre 1 y 15 a partir de ahora. Y así, el mejor modelo BPROP con este grupo de variables se alcanza con 13 nodos. En concreto, probamos entre 10 y 15 nodos, para ver que el mejor modelo BPROP se logra con 14 nodos. Aunque como se observa en la gráfica de abajo, aún es peor que los obtenidos en regresión lineal, que recordemos tenía un error promedio de en torno a 3.25x10^14.

A partir de ahora, únicamente tendremos en cuenta un máximo de 15 nodos; pues más de eso genera demasiado variabilidades y un error elevado. Realizamos un análisis en paralelo con el resto de algoritmos, y llegamos a la siguiente tabla:

Imagen 11: Proceso de elección de modelo de Redes Neuronales con variables del Modelo 4.



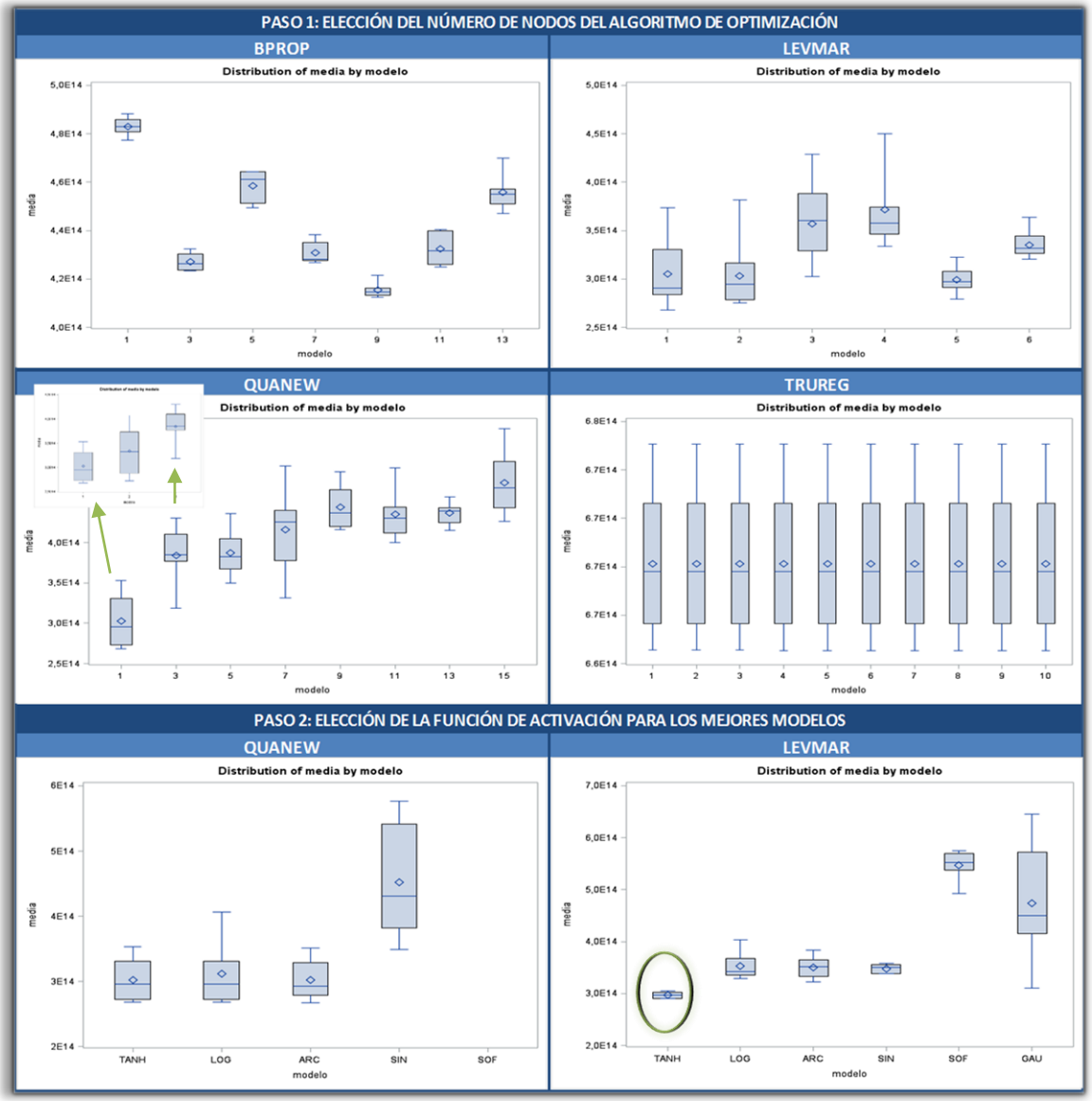
Como se observa en la gráfica, el mejor modelo de este set de variables es el formado por 3 nodos, con el algoritmo de optimización Levmar y la función de activación ARC. Sin embargo el promedio del error está por encima de 3,5 y por tanto no mejoramos el resultado de la regresión lineal.

4.4 ENSAYO CON MODELO DE 10 VARIABLES.

El siguiente grupo de variables con mejor resultado a nivel de validación cruzada en regresión lineal es el MODELO 3, que está compuesto únicamente por diez variables; de las cuales ocho son continuas (BUDG HT_IM OC WA_CR WA_N WA_UV YT_L YT_S), y dos son nominales (WA_UO, WA_CO). El bajo número de variables puede ser interesante, pues permitirá crear modelos más

complejos sin caer en el sobreajuste. A través de prueba y error, modificando los parámetros múltiples veces, llegamos al siguiente cuadro de resultados:

Imagen 12: Proceso de elección de modelo de Redes Neuronales con variables del Modelo 3.

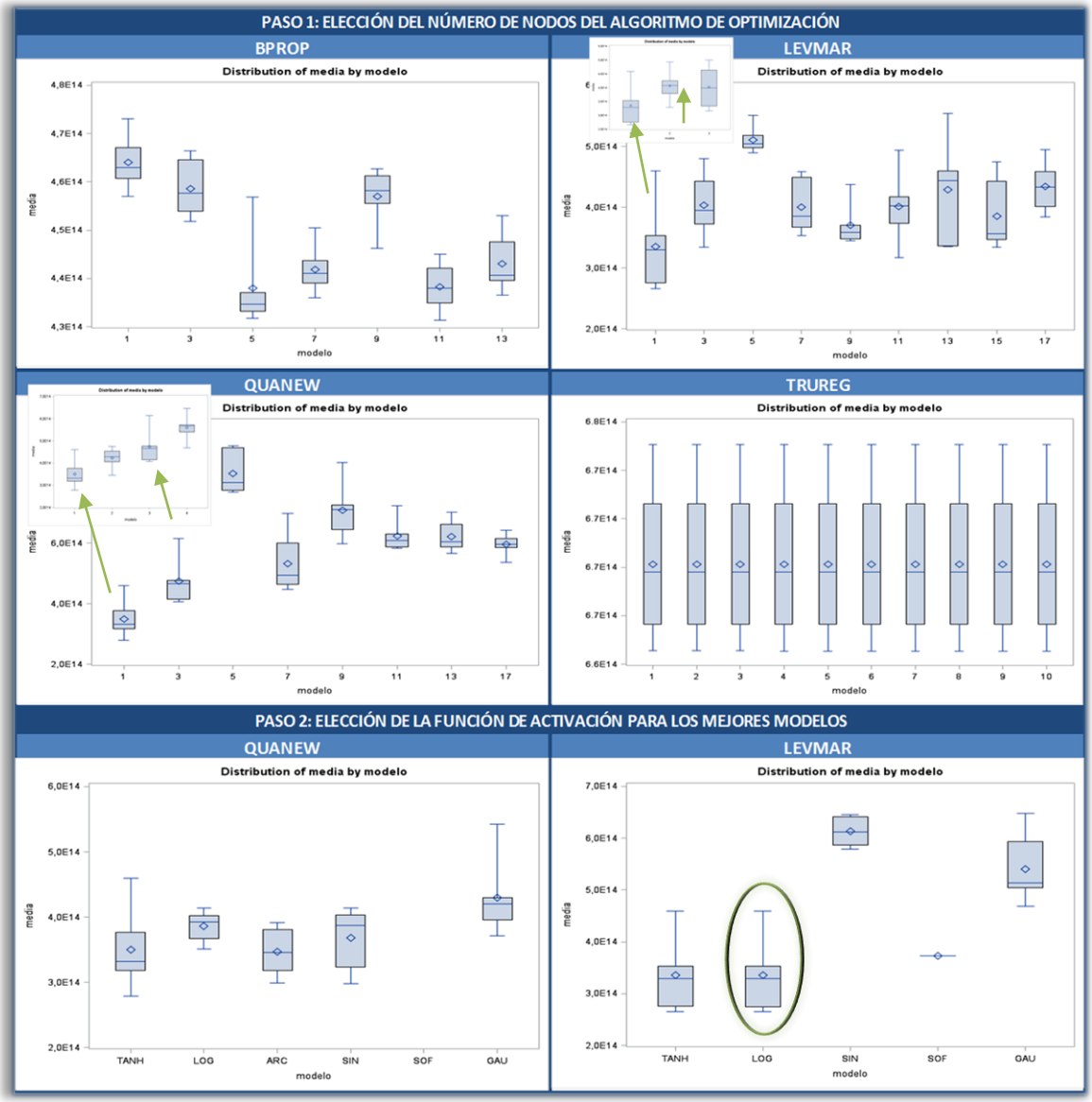


Como se puede observar en la tabla, existen algunos modelos de Levmar y de Quanew que se mantienen con un error en torno a 3x10^14. Si analizamos más en detallo los modelos de Levmar con 5 nodos y de Quanew con 1 nodo; vemos que son varias las funciones de activación que nos mantienen en el error anterior, y por tanto mejorando los resultados lineales. De entre ellos, sobresale por debajo de un error de 3x10^14; el modelo de Levmar con 5 nodos y función TANH..

4.5 ENSAYO CON MODELO DE 12 VARIABLES:

Por su parte el modelo cinco está formado por 12 variables; con un reparto de nueve variables continuas (DG C_N HT_IM OC WA_CR WA_N WA_UV YT_L YT_S) y tres variables nominales (WA_UO WA_CO WA_MM). A través de prueba y error, modificando los parámetros múltiples veces, llegamos al siguiente cuadro de resultados:

Imagen 13: Proceso de elección de modelo de Redes Neuronales con variables del Modelo 5.

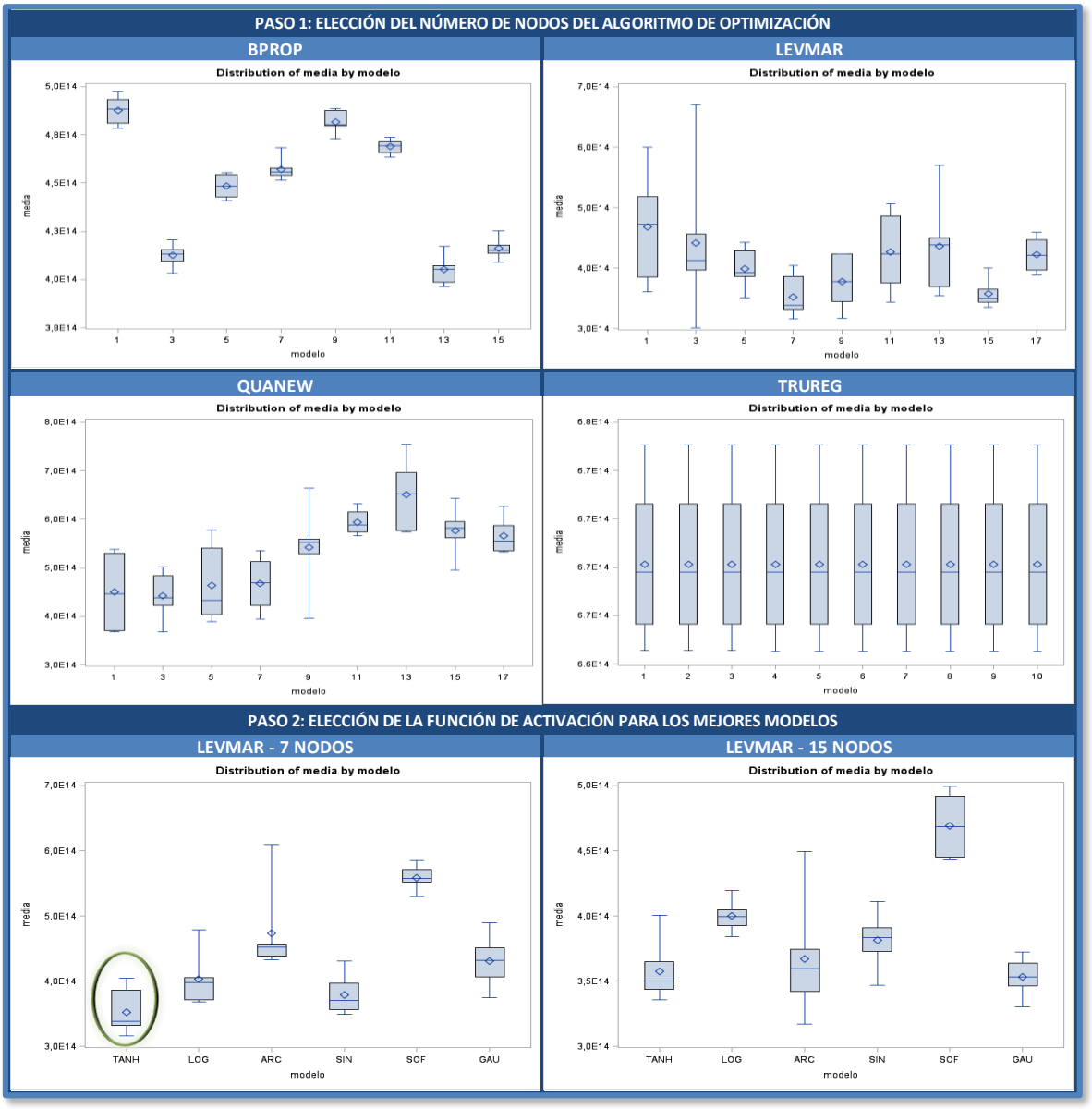


En este caso, de nuevo son los algoritmos de Levmar y Quanew los que mejor comportamiento presentan, pues se acercan a un error de 3x10^14. En concreto, y tras analizar la función de activación, vemos como son los modelos Levmar de 1 nodo, tanto con TANH como con LOG. Analizando en detalle, será LOG el que mejore muy levemente el resultado.

4.6 ENSAYO CON MODELO DE 13 VARIABLES.

Para terminar, analizamos el MODELO 6, que está formado por 13 variables; siendo ocho variables continuas (BUDG HT_IM OC WA_CR WA_N WA_UV YT_L YT_S) y cinco variables nominales (C_EU G_T WA_UO WA_CO WA_MM). A través de prueba y error, modificando los parámetros múltiples veces, llegamos al siguiente cuadro de resultados:

Imagen 14: Proceso de elección de modelo de Redes Neuronales con variables del Modelo 6



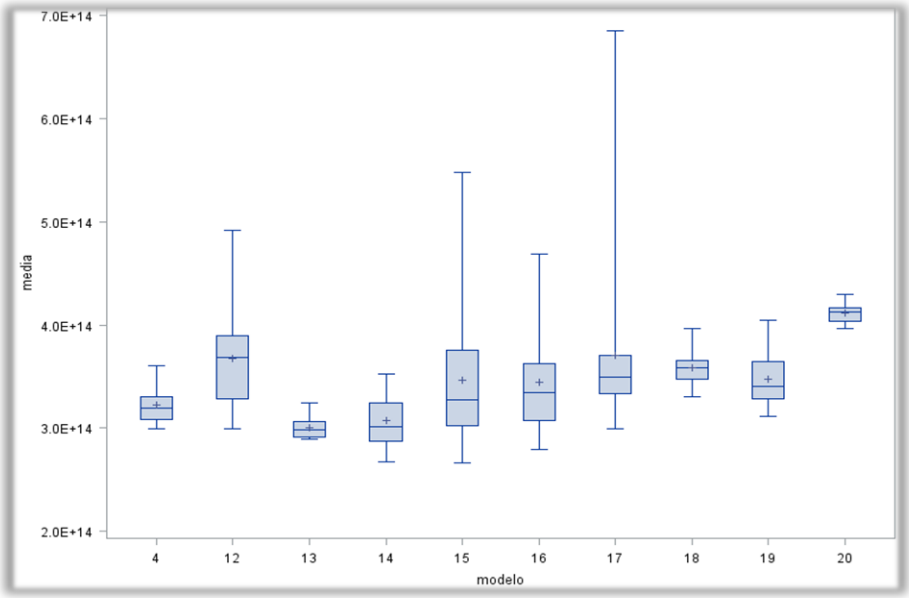
Como se puede observar, no entramos en el detalle unitario de los nodos, pues ya los primeros resultados nos muestran modelos con resultados peor valorados que cualquiera de los anteriores. Sin embargo, para seguir con el esquema de trabajo, desarrollamos la función de activación sobre Levmar de 7 y 15 nodos (por ser los únicos que bajan el error de los 4x10^14 puntos); y vemos que

el mejor modelo en este caso, es el formado por el algoritmo de optimización Levmar con 7 nodos y función de activación TANH.

4.7 COMPARACIÓN DE MODELOS MEDIANTE VALIDACIÓN CRUZADA.

Una vez que conocemos un número razonables de modelos, y los hemos analizado con validación cruzada para grupo de variables, pasamos a compararlos entre ellos. Sabemos que son al menos sub-óptimos; y para comprobar cuál es el mejor de ellos, realizamos validación cruzada sobre los mejores para determinar cuál es el mejor de todos ellos. Además, incluimos el modelo de regresión lineal que nos indica la bondad de este tipo de algoritmo para con nuestros datos. Así finalmente, obtenemos la siguiente gráfica:

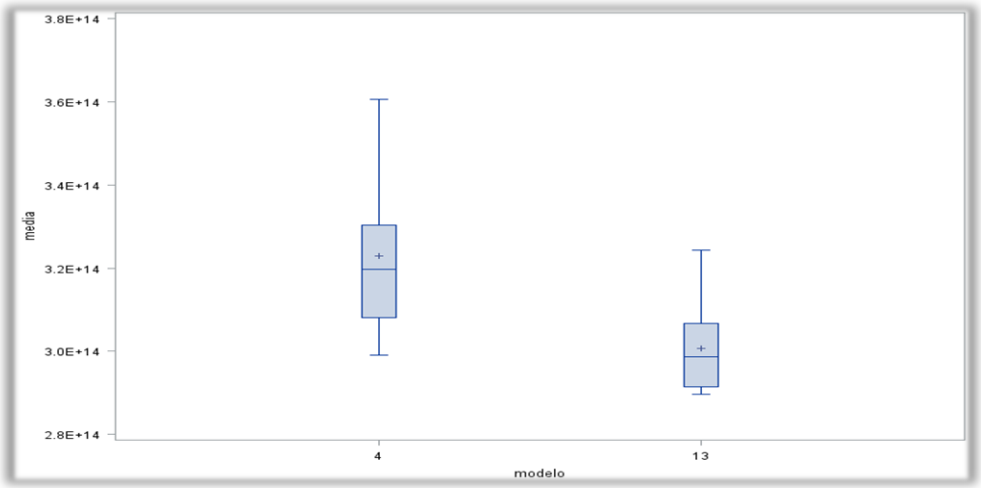
Imagen 15: Comparación Modelo Óptimo Lineal con Modelos de Redes



En la misma se observan tres modelos que sobresalen sobre el resto:

- Modelo número 4: Se trata del modelo creado con Regresión Lineal y determinante del primer set de variables analizado, y que contenía 15 variables.
- Modelo número 13: Este modelo fue el diseñado con el set de 10 variables analizado en segundo lugar, y se caracterizaba por tener cinco nodos con un algoritmo de optimización Levmar y una función de activación TANH.
- Modelo número 14: También este sobresale como uno de los mejores, pero siendo levemente peor que el trece al tener una mayor variabilidad del error, e incluso un error promedio muy levemente también más elevado.

Imagen 16: Comparación Modelo Óptimo Lineal y Óptimo de Redes



Si comparamos únicamente regresión lineal y redes, podemos comprobar como la segunda ha batido el resultado de la primera, colocando el nuevo error promedio en torno a 2,9x10^14. Para ello nos hemos servido únicamente de 10 de las 60 variables con las que contábamos. Éstas se han conseguido de (A) la parte de la base de datos con variables clásicas (BUDG y OC); (B) la parte temporal de IMDB obtenida a través de Web Archive (WA_CR, WA_N, WA_UV, WA_UO y WA_CO); (C) de YouTube (YT_L y YT_S) y, finalmente, (D) del término de búsqueda de twitter (HT_IM).

5 MACHINE LEARNING: MODELOS DE REGRESIÓN CON ALGORITMOS RANDOM FOREST.

Para entender el proceso que vamos a desarrollar a continuación, es necesario saber cuál es el concepto del algoritmo de árbol (tree algorithm en inglés). Se trata de una estructura de datos abstracta y generalmente no lineal; que simula una jerarquía en forma de árbol; es decir, es una colección de nodos con una raíz, y una concatenación de nodos padres y sus consecuentes subárboles en diferentes niveles.

Estos modelos de clasificación son por lo general muy inconsistentes. Para incrementar la precisión y la estabilidad de los árboles, surge en 1994 un algoritmo de ensamble machine learning llamado Bootstrap aggregating o Bagging. Este algoritmo consiste en interrelacionar múltiples árboles, de forma que los errores de unos se promedien unos con otros hasta lograr un algoritmo más robusto.

(Leo Breiman, 1994)

Ahora bien, este trabajo que aquí se presenta, decide analizar la siguiente evolución de los árboles, Random forest[6] (en español “Selvas Aleatorias”). Este algoritmo, desarrollado en 2001 por Leo

[66] Random Forest- Wikipedia: https://en.wikipedia.org/wiki/Random_forest

Breiman y Adele Cutler es una combinación de árboles predictores tal que cada árbol depende de los valores de un vector aleatorio probado independientemente y con la misma distribución para cada uno de estos. Es una modificación sustancial de bagging que construye una larga colección de árboles no correlacionados y luego los promedia. Es decir, el método combina la idea de bagging y la selección aleatoria de atributos, introducida para construir una colección de árboles de decisión con variación controlada. Este método es popular y ampliamente utilizado, pues si bien tiene problemas similares a otros como boosting (que desarrollaremos en el punto siguiente), es más fácil de entrenar y ajustar. (Breiman, 2001) ("Random forest - Wikipedia, la enciclopedia libre," 2015).

Para no desarrollar de manera tediosa cada uno de los pasos llevados a cabo en la optimización del algoritmo, vamos a explicar cuáles son los parámetros que hemos ido controlando, así como los pasos seguidos para intentar comprender como cada uno de esos parámetros afectan en el resultado final del error. Posteriormente analizaremos los resultados más relevantes que hemos obtenido (hay que tener en cuenta que, de cada modelo mostrado, al menos existen 5 pruebas diferentes por detrás).

5.1 PARÁMETROS A AJUSTAR

Pasamos ahora a definir cuáles son los parámetros más relevantes, y como se incluyen dentro de la macro para finalmente poder llegar a aplicar un código de prueba como el que sigue:

Código 6: Ejemplo de codificación de macro en Random Forest.

```
%cruzadarandomforest(archivo=local.masterdb,vardep=y,listconti=BUDG
HT_IM OC WA_CR WA_N WA_UV YT_L YT_S,
listcategor=WA_UO WA_CO ,
maxtrees=100,variables=3,porcenbag=0.80,maxbranch=2,tamhoja=5,maxdepth=
10,pvalor=0.1,
ngrupos=4,sinicio=12345,sfinal=12355);
```

Grupo de Variables: Al igual que en redes, las 60 variables iniciales hemos realizado un control de las más relevantes de cara a estimar la variable dependiente. En este caso vamos a trabajar con dos grupos. En primer lugar, probaremos las 10 variables que mejor resultado nos arrojó en redes. Y posteriormente realizaremos un análisis paralelo sobre las 15 variables que aparecen más veces en los diferentes modelos de Regresión Lineal que hemos probado, pues los árboles por regla general permitirán incorporar más variables sin caer en sobreajuste (como si pasaba en las redes).

Código 7: Ejemplo de codificación de la elección de las variables en Random Forest.

```
vardep=y,listconti=BUDG HT_IM OC WA_CR WA_N WA_UV YT_L YT_S,
listcategor=WA_UO WA_CO ,
```

Semilla: La comprobación de la bondad de cada modelo, la realizaremos mediante validación cruzada. Para mantener la coherencia con los otros algoritmos desarrollados, seguiremos utilizando 15 semillas, desde la 12345 hasta la 12360; realizando los cálculos en cuatro grupos.

Código 8: Ejemplo de codificación de los grupos y semilla en validación cruzada para Random Forest

```
ngrupos=4,sinicio=12345,sfinal=12355
```

MaxTree: Especifica el número de árboles a promediar en el modelo forest.

Código 9: Ejemplo de codificación del número máximo de árboles en Random Forest

```
maxtrees=&maxtrees
```

Vars_to_try: Número de variables input que se van a dividir en un nodo. Es decir, de las 15 o 10 variables (según el modelo), el algoritmo no tiene por qué escoger todas, sino que cogerá el número que indiquemos de manera aleatoria para clasificar los nodos que se creen en el árbol.

Código 10: Ejemplo de codificación del número de variables en Random Forest.

```
vars_to_try=&variables
```

Trainfraction: Especifica la fracción de observaciones de entrenamiento con las que se van a entrenar el árbol y c modelo.

Código 11: Ejemplo de codificación del porcentaje de entrenamiento en Random Forest.

```
trainfraction=&porcenbag
```

MaxBranch: Restringe el número de subgrupos que una regla específica de división puede crear.

Código 12: Ejemplo de codificación del número de ramas por nodo en Random Forest.

```
MAXBRANCH=n
```

LeafSize: Especifica el número mínimo de observaciones de entrenamiento que puede tener una nueva rama para ser creada.

Código 13: Ejemplo de codificación del tamaño de las hojas en Random Forest.

```
leafsize=&tamhoja
```

MaxDepth: Especifica el número máximo de generaciones de nodos. Al nodo original, generación 0, se le llama nodo raíz. Los hijos del nodo raíz son de primera generación.

Código 14: Ejemplo de codificación de la profundidad en Random Forest.

```
maxdepth=&maxdepth
```

Alpha: Especifica el p-valor usado para probar la asociación entre una variable candidata y la variable objetivo. Si ninguna asociación cumple este umbral, el nodo no se divide.

Código 15: Ejemplo de codificación del ajuste del PValor en Random Forest.

```
alpha=&pvalor
```

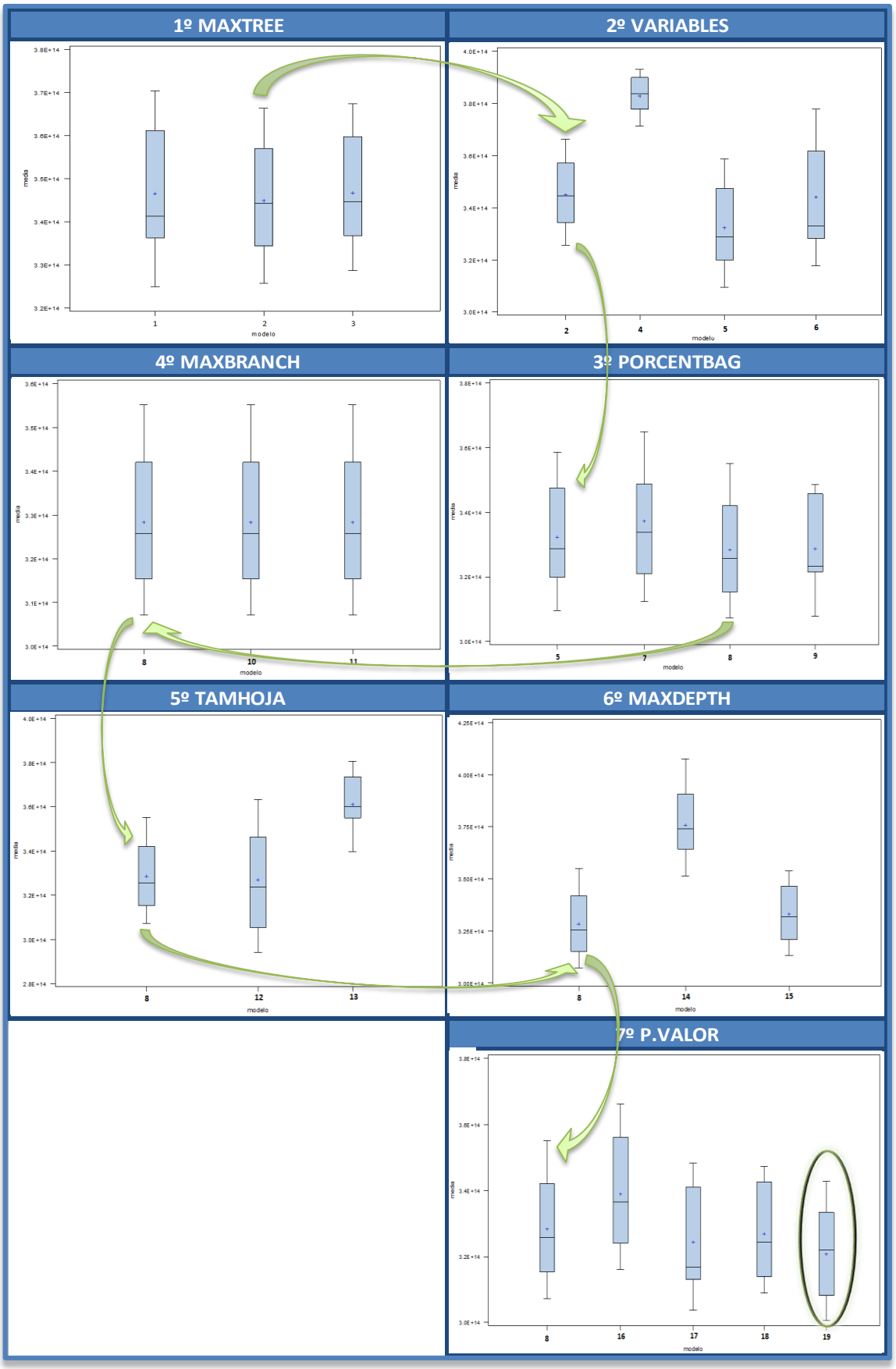
5.2 PROCESO DE OPTIMIZACIÓN CON VARIABLES DE REDES.

Dado el amplio número de parámetros a modificar, debemos realizar un proceso que nos lleve al mejor resultado posible sin tener que analizar cada una de las opciones; y es que, para ello, si únicamente analizáramos 4 valores de cada componente, deberíamos realizar más de 2.000 modelos, para cada set de variables. Así, lo que realizamos es ajustar cada uno de los parámetros, para después pasar a optimizar el resto suponiendo que el primeramente ajustado es recomendable:

Tabla 9: Proceso de elección del modelo óptimo en Random Forest.

Rforest	:	MaxTree		Variables		PorcentBag		MaxBranch		TamHoja		MaxDepth		Pvalor	
PRUEBA	1	100	3º	3		0,8		2		5		10		0,1	
PRUEBA	**2**	**200**	**1º**	3	1º	0,8		2		5		10		0,1	
PRUEBA	3	300	2º	3		0,8		2		5		10		0,1	
PRUEBA	4	200		1	4º	0,8		2		5		10		0,1	
PRUEBA	5	200		**6**	**1º**	0,8	3º	2		5		10		0,1	
PRUEBA	6	200		8	3º	0,8		2		5		10		0,1	
PRUEBA	7	200		6		0,9	4º	2		5		10		0,1	
PRUEBA	8	200		6		**0,7**	**1º**	**2**	**1º**	**5**	**1º**	**10**	**1º**	0,1	4º
PRUEBA	9	200		6		0,6	2º	2		5		10		0,1	
PRUEBA	10	200		6		0,7		3	2º	5		10		0,1	
PRUEBA	11	200		6		0,7		4	3º	5		10		0,1	
PRUEBA	12	200		6		0,7		2		2	2º	10		0,1	
PRUEBA	13	200		6		0,7		2		12	3º	10		0,1	
PRUEBA	14	200		6		0,7		2		5		3	3º	0,1	
PRUEBA	15	200		6		0,7		2		5		20	2º	0,1	
PRUEBA	16	200		6		0,7		2		5		10		0,05	5º
PRUEBA	17	200		6		0,7		2		5		10		0,2	2º
PRUEBA	18	200		6		0,7		2		5		10		0,15	3º
PRUEBA	19	200		6		0,7		2		5		10		**0,25**	**1º**

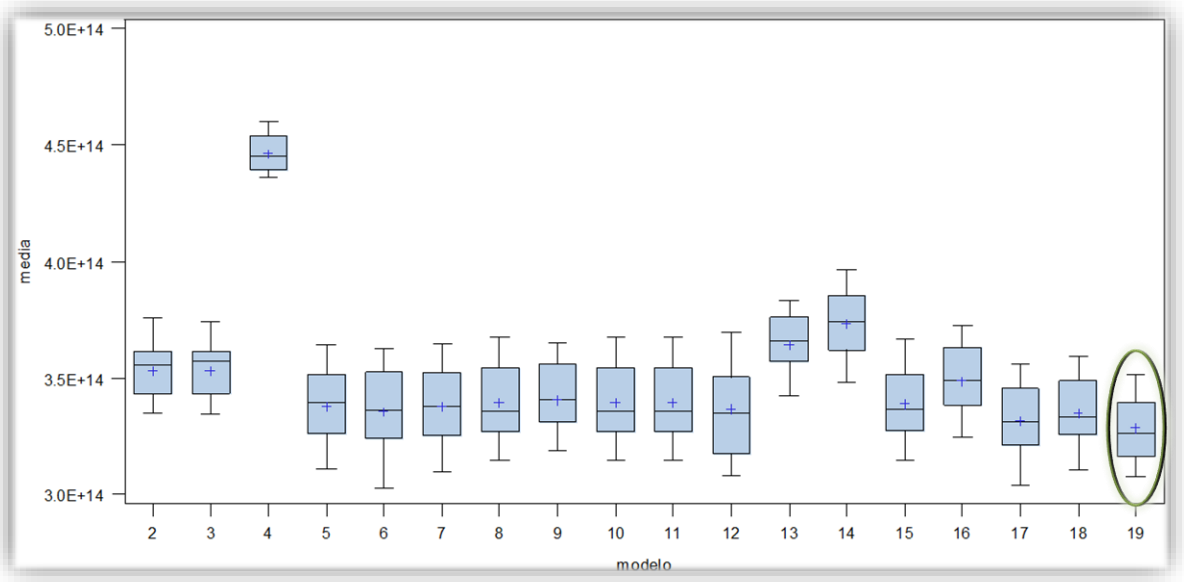
Imagen 17: Proceso de optimización de Random Forest con variables de redes neuronales.


1º MAXTREE
2º VARIABLES
4º MAXBRANCH
3º PORCENTBAG
5º TAMHOJA
6º MAXDEPTH
7º P.VALOR
media
modelo

5.3 PROCESO DE OPTIMIZACIÓN DE RANDOM FOREST CON VARIABLES DE REGRESIÓN LINEAL

De la misma forma que realizamos el estudio con las variables que mostraron un mejor comportamiento en redes; también realizamos un estudio incluyendo las 15 mejores variables de la regresión lineal que nos sirvió para escoger esos grupos de variables en redes.

Imagen 18: Proceso de optimización de Ramdom Forest con variables de regresión lineal.



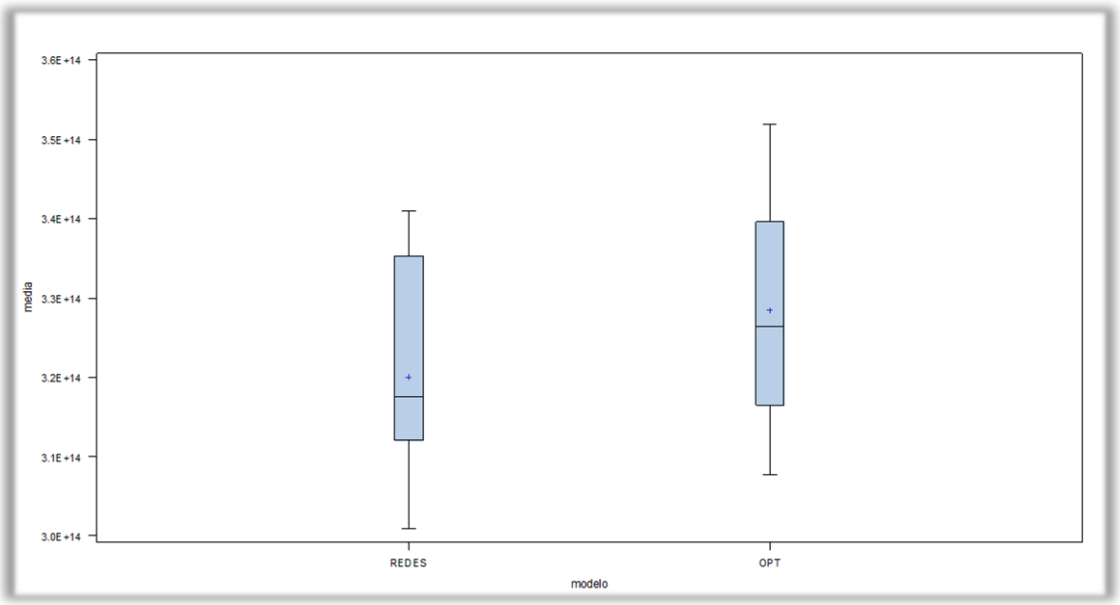
5.4 ELECCIÓN DEL MEJOR MODELO

Los dos mejores modelos obtenidos en Random Forest, han sido formados de la misma manera, la única diferencia es el set de variables que determinan como se dividen los árboles.

El modelo se ha desarrollado con los siguientes parámetros: Un promedio de 200 árboles, escogiendo hasta 6 variables para realizar la partición en la clasificación de las ramas. Se utiliza el 70% de las observaciones para realizar ese modelo como formación *train*. La apertura de los nodos se realiza de dos en dos. Y el árbol estña formado por hojas de, al menos, tamaño 5 y una profundidad máxima de 10. EL P-Valor mejora cuanto mayor es, hasta llegar a una curva de ruptura de tendencia en 0.4 puntos.

Como se puede observar, ha sido el modelo calculado con las variables utilizadas en redes el que porporciona el mejor resultado.

Imagen 19: Selección del mejor modelo de Redes Neuronales.



6 MACHINE LEARNING: MODELOS DE REGRESIÓN CON ALGORITMOS GRADIENT BOOSTING

Boosting[7] es un algoritmo de *machine learning* creado en 1990 por Robert Schapire. Este se basa en ensamblado, y su utilidad principal es reducir la varianza de un conjunto de "weak learners" (algoritmos con poca correlación con la variable dependiente) en "strong learners" (algoritmos robustos con mayor capacidad de predicción debido a la fuerte correlación con la variable dependiente). (Michael Kearns, 1988)

En palabras de Robert Schapire: *"Informally, [the hypothesis boosting] problem asks whether an efficient learning algorithm […] that outputs a hypothesis whose performance is only slightly better than random guessing [i.e. a weak learner] implies the existence of an efficient algorithm that outputs a hypothesis of arbitrary accuracy [i.e. a strong learner]."* (Schapire, 1990)

Por tanto, en los años 90 nos encontramos con la hipótesis del problema boosting, "*hypothesis boosting problem* ", que transforma un grupo de modelos poco correlados, en otro de mayor correlación a través de su ensamble. Y posteriormente los algoritmos que lograban satisfacer dicha hipótesis, se denominaron como "*BOOSTING*". Por su parte, también puede encontrarse bajo el nombre ARCING (Adaptative Resampling and Combining), desarrollado por *Freund y Schapire*.

[7] Gradient Boosting- Wikipedia: https://en.wikipedia.org/wiki/Gradient_boosting

(Schapire, 2003) Al igual que ocurrió con el punto anterior, para no desarrollar de manera tediosa cada uno de los .pasos llevados a cabo en la optimización del algoritmo, vamos a explicar los parámetros controlados y el proceso seguido en su optimización del error promedio y varianza. Posteriormente analizaremos los resultados más relevantes que hemos obtenido (hay que tener en cuenta que, de cada modelo mostrado, al menos existen 5 pruebas diferentes por detrás). (Denil et al., 2013)

6.1 PARÁMETROS A AJUSTAR

Pasamos ahora a definir cuáles son los parámetros más relevantes, y como se incluyen dentro de la macro para finalmente poder llegar a aplicar un código de prueba como el que sigue:

Código 16: Ejemplo de aplicación de macro en Gradient Boosting.

```
cruzadatreeboost(archivo=local.masterdb,  vardepen=y
                 ,conti=BUDG HT_IM OC WA_CR WA_N WA_UV YT_L YT_S
                 ,categor=WA_UO WA_CO
                 ,ngrupos=4  ,sinicio=12345    ,sfinal=12360
                 ,leafsize=25, iteraciones=300, shrink=0.1
                 ,maxbranch=2, maxdepth=4, mincatsize=1, minobs=10);
data final21;set final;modelo=21;
```

Grupo de Variables: Al igual que en redes, las 60 variables iniciales hemos realizado un control de las más relevantes de cara a estimar la variable dependiente. En este caso vamos a trabajar con dos grupos. En primer lugar, probaremos las 10 variables que mejor resultado nos arrojó en redes. Y posteriormente realizaremos un análisis paralelo sobre las 15 variables que aparecen más veces en los diferentes modelos de Regresión Lineal que hemos probado, pues los árboles por regla general permitirán incorporar más variables sin caer en sobreajuste (como si pasaba en las redes).

Semilla: La comprobación de la bondad de cada modelo, la realizaremos mediante validación cruzada. Para mantener la coherencia con los otros algoritmos desarrollados, seguiremos utilizando 15 semillas, desde la 12345 hasta la 12360; realizando los cálculos en cuatro grupos.

Iterations: Especifica el número de elementos en las series incrementales. Para variables objetivo binarias o intervalo (nuestro caso), el número de iteraciones es igual que el de árboles. Para una variable objetivo nominal se creará un árbol separado para cada categoría de la variable objetivo en cada iteración.

Código 17: Ejemplo de codificación del número de iteraciones en Gradient Boosting.

```
iterations=&iteraciones
```

Shrinkage: También llamado parámetro de regularización, se trata de un parámetro estadístico que consiste en modificar el ratio de aprendizaje. Empíricamente se ha demostrado que generalmente un ratio pequeño mejora la correlación del modelo, pero a un mayor coste computacional, pues a menor ratio de aprendizaje, mayor será el número de iteraciones.

Esto se basa en la fórmula: $F_m(x) = F_{m-1}(x) + \nu \cdot \gamma_m h_m(x), \quad 0 < \nu \leq 1,$ donde v es el ratio de aprendizaje.

Código 18: Ejemplo de codificación del parámetro Shrinkage en Gradient Boosting.

```
shrinkage=&shrink
```

LeafSize: Especifica el número menor de observaciones de entrenamiento que puede tener una nueva rama para ser creada.

Código 19: Ejemplo de codificación del número de LeafSize en Gradient Boosting.

```
leafsize=&leafsize
```

MaxBranch: Restringe el número de subconjuntos que una regla de división puede crear en el número especificado o menos. Por ejemplo, un valor de 2 da como resultado árboles binarios.

Código 20: Ejemplo de codificación del número de MaxBranch en Gradient Boosting.

```
maxbranch=&maxbranch
```

MaxDepth: Especifica el número máximo de generaciones de nodos. Al nodo original, generación 0, se le llama nodo raíz. Los hijos del nodo raíz son de primera generación.

Código 21: Ejemplo de codificación de MaxDepth en Gradient Boosting.

```
maxdepth=&maxdepth
```

MinCatSeize: Número mínimo de observaciones para un valor categórico. Una categoría debe aparecer en al menos el número de observaciones especificado para utilizarlo en la división.

Código 22: Ejemplo de codificación del tamaño minimo de las categorías en Gradient Boosting.

```
mincatsize=&mincatsize
```

MinObs: Especifica el número más pequeño de observaciones de entrenamiento que un nodo puede tener para que el procedimiento considere dividirlo.

Código 23: Ejemplo de codificación del mínimo de observaciones en Gradient Boosting.

```
splitsize=&minobs
```

6.2 VARIABLES DEL MEJOR MODELO DE REDES.

Este conjunto de variables está formado por 10 de las 60 variables con las que contamos. De ellas, ocho son continuas (BUDG HT_IM OC WA_CR WA_N WA_UV YT_L YT_S) y dos son nominales (WA_UO WA_CO)

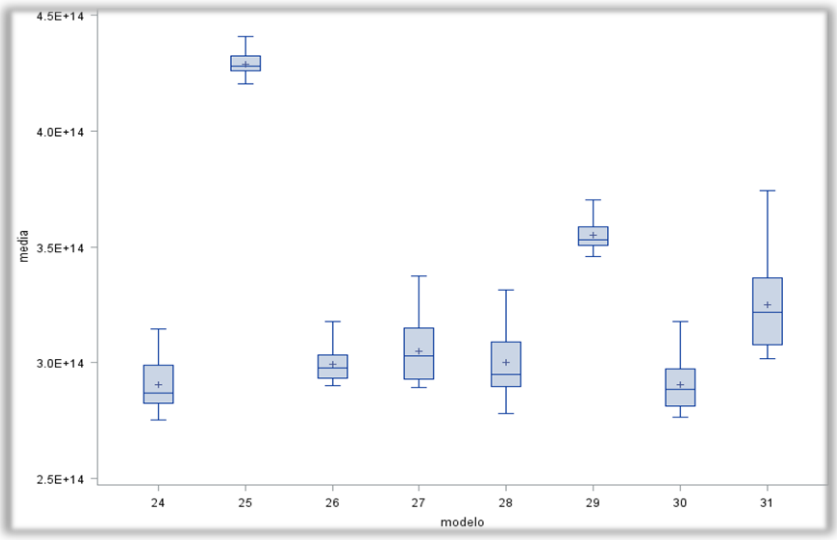
Al igual que en Random Forest, el número de parámetros a modificar es relativamente amplio. Por ello no podemos aplicar Dado el amplio número de parámetros a modificar, debemos realizar un proceso que nos lleve al mejor resultado posible sin tener que analizar cada una de las opciones; y es que, para ello, si únicamente analizáramos 4 valores de cada componente, deberíamos realizar más de 2000 modelos, para cada set de variables. Así, lo que realizamos es ajustar cada uno de los parámetros, para después pasar a optimizar el resto suponiendo que el primeramente ajustado es recomendable. Hasta el momento, este set de variables se está mostrando como el más acertado a la hora de estimar los ingresos del primer fin de semana de películas comerciales en EEUU.

En primer lugar, realizamos un análisis del parámetro de iteraciones, con valores desde los 100 hasta los 800. Como 100 es el número que mejor se comporta, añadimos 110 y 90; y confirmamos que 100 es, en principio, un número acertado de iteraciones, pues ambos se comportan peor.

Posteriormente probamos el parámetro Shrink, sabemos que este valor debe ser pequeño para ajustar mejor los modelos, por lo que realizamos pruebas desde los 0,5 hasta 0,01. Comprobamos que valores tan pequeños como 0,01 empeoran el error promedio, y que los mejores valores se encuentran entre 0,1 y 0,05.

Conociendo estos valores realizamos una prueba general con aquellos que se han portado mejor, así incluimos 100 iteraciones (Shrink 0,1 – 0,01 – 0,05- 0,2) y lo mismo con 200 iteraciones.

Imagen 20: Proceso cálculo Gradient Boosting con variables de Redes



Como se observa, hay dos modelos parecidos (24 y 30) El resultado es interesante, pues el modelo 24 era nuestro mejor modelo mirando los parámetros por separado, pero ahora el modelo 30 muestra un error similar, por lo que también lo estudiaremos.

De la misma forma que para esos dos parámetros, se realizan permutaciones sobre el resto de variables, y en ningún caso somos capaces de mejorar el modelo.

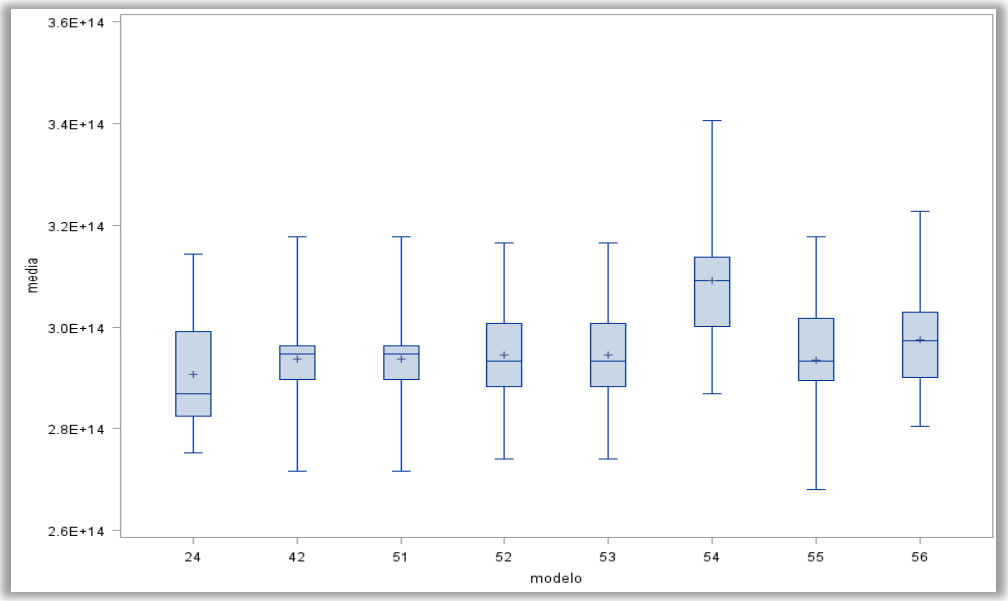
Por lo tanto, para este set de variables, podemos identificar el mejor modelo como: leafsize=25, iteraciones=100, shrink=0.1, maxbranch=2, maxdepth=4, mincatsize=15, minobs=10.

6.3 VARIABLES DEL MEJOR MODELO DE REGRESIÓN LINEAL

Pasamos ahora a analizar otro set de variable que guarda correlación con la variable dependiente. Son 15 variables, de las que 12 son continuas (BUDG HT_IM OC WA_CR WA_N WA_UV YT_L YT_S YT_V C_N AS_NT WA_UC LET) y 3 nominales (WA_UO WA_CO YEAR).

Realizamos un circuito similar, por el que comenzamos analizando los parámetros de iteraciones y shrink; y llegamos a la conclusión que los mejores modelos están formados por 100 iteraciones y un valor Shrink de 0,05. A partir de ahí, el LeafSize si nos aporta una mejora en nuestro modelo, según el valor utilizado. Los restantes no mejoraban sustancialmente los resultados. Por lo tanto, llegamos a la conclusión que nuestro mejor con este set de variables (el número 42 en la gráfica inferior) está formado por: leafsize=25, iteraciones=100, shrink=0.1, maxbranch=2 , maxdepth=4, mincatsize=15, minobs=10).

Imagen 21: Proceso cálculo Gradient Boosting con variables de R.Lineal



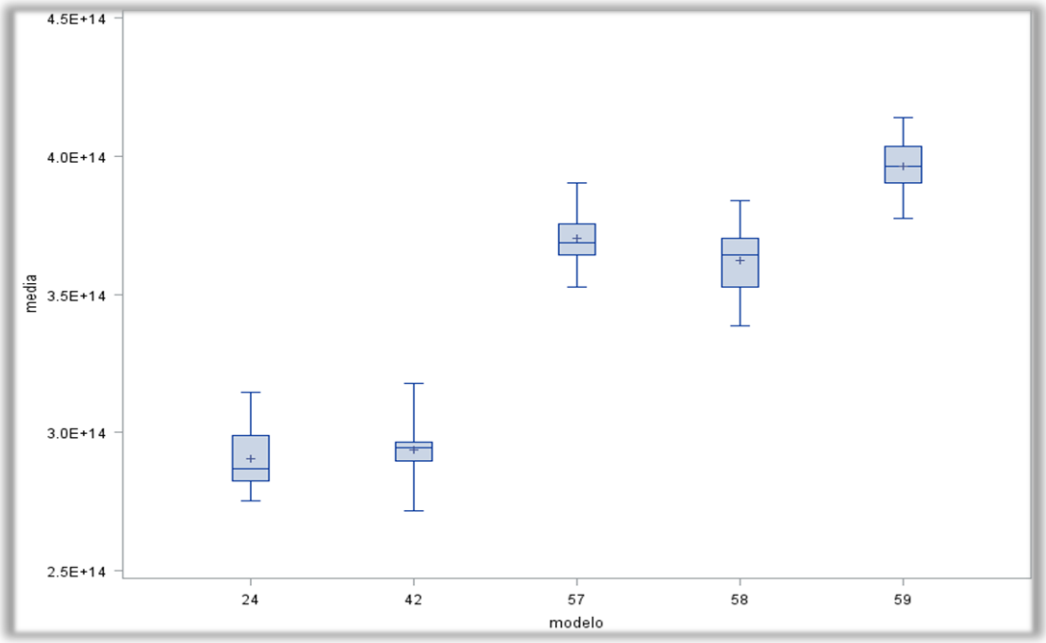
Como se observa en la gráfica, si se compara con el modelo 24 (mejor modelo del set de variables de redes); el resultado es similar. A pesar de que la variabilidad total es mayor, la caja es mucho más pequeña lo que nos muestra que la variabilidad quitando los extremos es mejor. Además, la media está relativamente más alta, pero el fin de la caja está más abajo. Por lo que tendremos en cuenta este modelo a la hora de realizar análisis posteriores.

6.4 OTRAS PRUEBAS

Dados los buenos resultados obtenidos con este algoritmo, escogemos dos sets de variables extras para hacer pruebas sobre los parámetros que sabemos que funcionan relativamente bien.

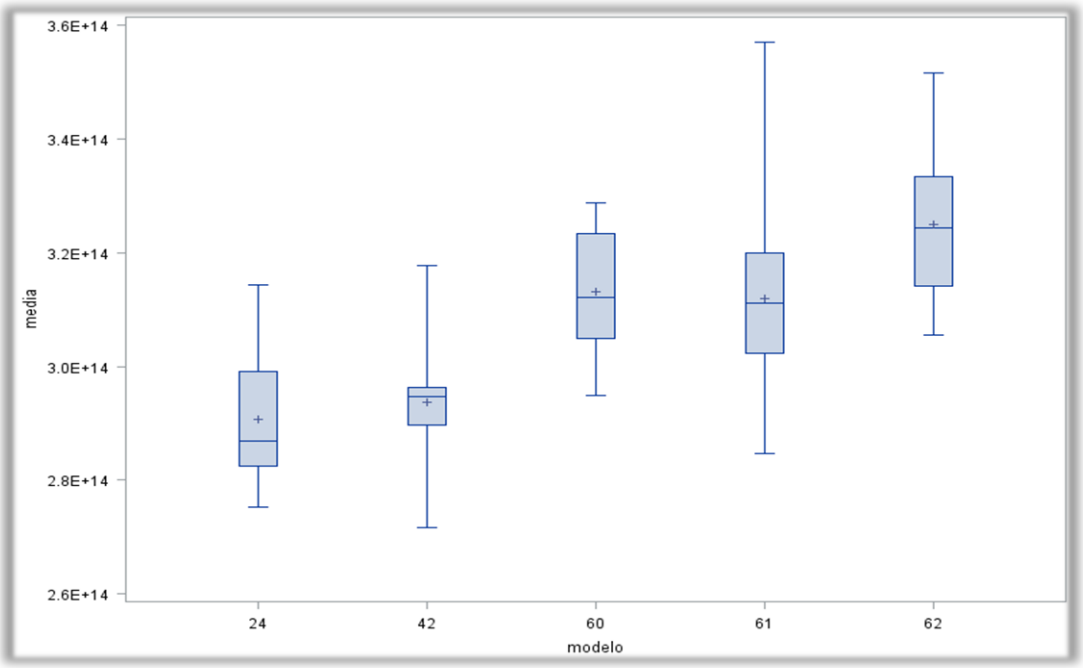
En primer lugar, sacamos un grupo de variables que en el análisis individual del punto 3, nos pareció que se podían ajustar bien a los modelos desarrollados, pero que finalmente éstos no contemplaron. Así creamos una serie de modelos con 26 variables, de las cuales 14 son nominales y 12 son continuas. Como se observa en la gráfica el resultado fue poco relevante.

Imagen 22: Proceso cálculo Gradient Boosting con variables lógicas



En segundo, sabiendo que los árboles realizan una selección de variables bastante potente, introducimos todas con las que contamos para ver si los resultados pudieran mejorar en algún caso. El resultado reflejado en la tabla inferior, de nuevo nos muestra una menor correspondencia que la hallada con los modelos de las variables seleccionadas en redes y en lineal.

Imagen 23: Proceso cálculo Gradient Boosting con todas las variables dependientes



7 COMPARACIÓN DE MODELOS

Los modelos óptimos de cada tipo de algoritmos son:

REGRESIÓN LINEAL (RL):

Código 24: Ejemplo de codificación del número de iteraciones en Gradient Boosting.

```
%cruzada(archivo=local.masterdb
      ,vardepen=Y
      ,conti=AS_NT BUDG HT_IM OC WA_CR WA_N WA_UV YT_L YT_S
      ,categor=L_E C_EU G_T WA_UO WA_CO WA_MM
      ,ngrupos=4, sinicio=12345, sfinal=12360);
data final1;set final;modelo='RL';
```

REDES NEURONALES (RN):

Código 25: Ejemplo de codificación del número de iteraciones en Gradient Boosting.

```
%cruzadaneural(archivo=local.masterdb
      ,vardepen=Y
      ,conti=BUDG HT_IM OC WA_CR WA_N WA_UV YT_L YT_S
      ,categor=WA_UO WA_CO
      ,ngrupos=4, sinicio=12345, sfinal=12360
      ,ocultos=5, meto=LEVMAR ,acti=TANH);
data final2;set final;modelo='RN';
```

RANDOM FOREST (RF):

Código 26: Ejemplo de codificación del número de iteraciones en Gradient Boosting.

```
%cruzadarandomforest(archivo=local.masterdb,vardep=y,
      listconti=BUDG HT_IM OC WA_CR WA_N WA_UV YT_L YT_S,
      listcategor=WA_UO WA_CO ,
      maxtrees=200,variables=6,porcenbag=0.70,maxbranch=2,tamhoja=5,max
      depth=10,pvalor=0.4, ngrupos=4,sinicio=12345,sfinal=12360);
data final3;set final;modelo='RF';
```

GRADIENT BOOSTING – VARIABLES DE REDES (GBr/rG):

Código 27: Ejemplo de codificación del número de iteraciones en Gradient Boosting.

```
%cruzadatreeboost(archivo=local.masterdb
      ,vardepen=y
      ,conti=BUDG HT_IM OC WA_CR WA_N WA_UV YT_L YT_S
      ,categor=WA_UO WA_CO
      ,ngrupos=4, sinicio=12345, sfinal=12360
      ,leafsize=25, iteraciones=100, shrink=0.1
      ,maxbranch=2, maxdepth=4, mincatsize=15, minobs=10);
data final4;set final;modelo='rGB';
```

GRADIENT BOOSTING – VARIABLES LINEALES (GBl/lG):

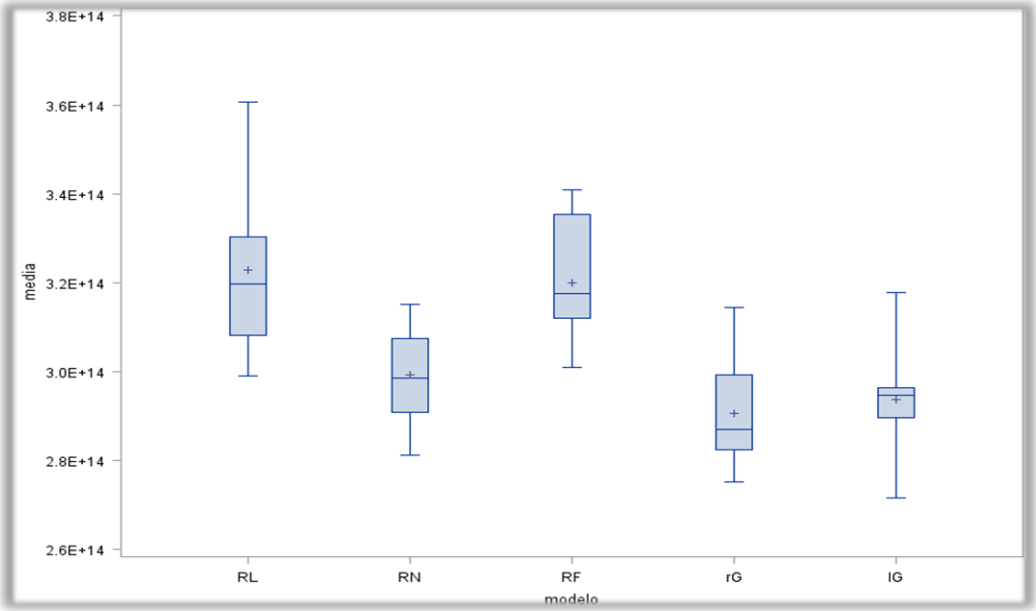
Código 28: Ejemplo de codificación del número de iteraciones en Gradient Boosting.

```
%cruzadatreeboost(archivo=local.masterdb
      ,vardepen=y
      ,conti=BUDG HT_IM OC WA_CR WA_N WA_UV YT_L YT_S YT_V C_N AS_NT
             WA_UC LET
      ,categor=WA_UO WA_CO YEAR
      ,ngrupos=4,sinicio=12345,sfinal=12360
      ,leafsize=25,iteraciones=100,shrink=0.1,maxbranch=2
      ,maxdepth=4 ,mincatsize=15,minobs=10);
data final5;set final;modelo='lGB';
```

Después de analizar cuatro algoritmos diferentes para ver qué método es el más adecuado a la hora de predecir nuestra variable dependiente, llegamos a la conclusión de que *gradient boosting* funciona mejor que el resto; siendo el segundo mejor modelo el de redes neuronales. Mientas que la regresión lineal y *random forest* no se comportan tan bien.

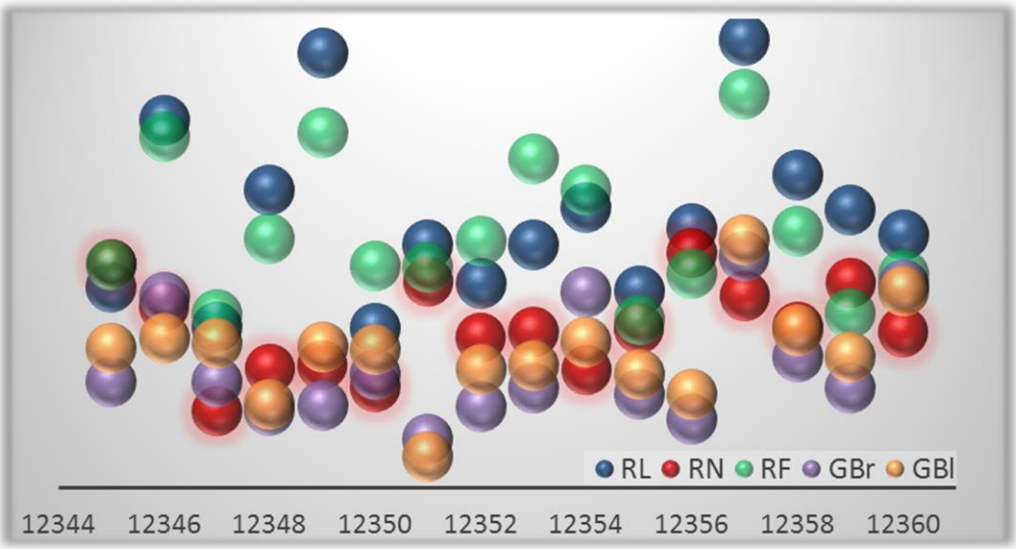
En los anexos podemos ver una tabla 1 con los mismos datos de la gráfica, pero distinguiendo valor a valor.

Imagen 24: Comparación final modelos óptimos por tipo de algoritmo.



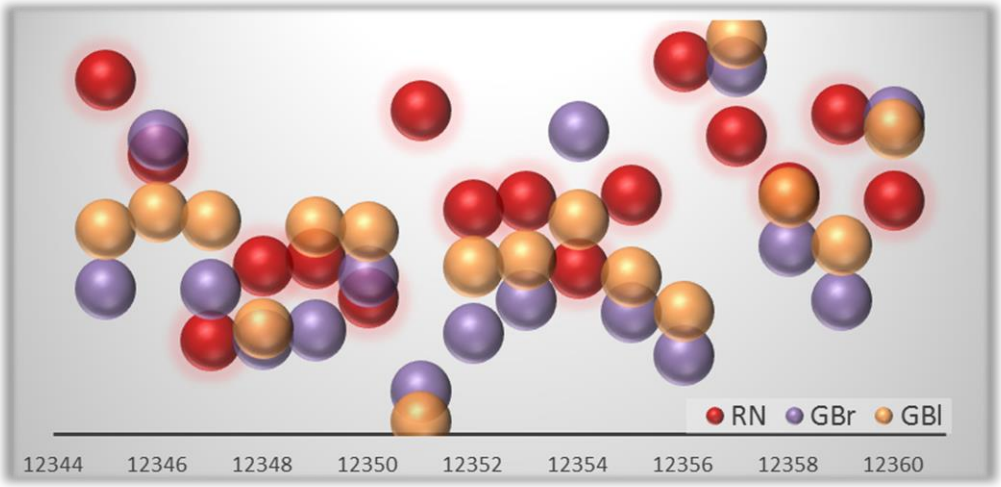
Generando la gráfica del error promedio obtenido para cada semilla en cada uno de los algoritmos (véase imagen 25) podemos apreciar como los algoritmos de *gradient boosting* y redes neuronales se comportan mejor, resultando en modelos más precisos.

Imagen 25: Error promedio de los modelos óptimos por algoritmo



Eliminando de la gráfica aquellos algoritmos que se comportan claramente peor que el resto (regresión lineal y *random forest*), podemos apreciar como los restantes responden de manera similar a nuestro conjunto de datos.

Imagen 26: Error promedio de los 3 mejores modelos finales

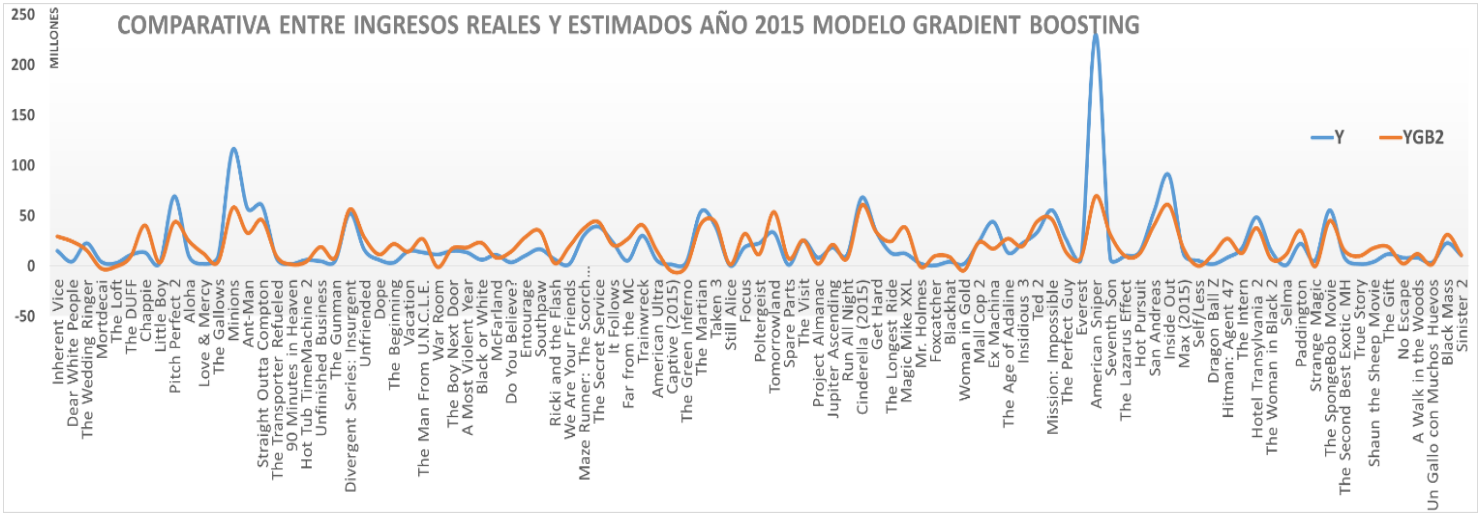


8 RESULTADOS Y CONCLUSIONES

Una vez obtenido nuestro mejor modelo, pasamos a aplicarlo sobre los resultados de prueba de 2015, obteniendo los siguientes resultados:

En primer lugar, realizamos una comparación, película a película de la variable real (Y), y la variable estimada (YGB2-Gradient Boosting segundo óptimo con variables lineales); como se puede observar en la imagen 27, el modelo es más preciso a la hora de predecir las películas dentro de un rango normal, aunque no se muestra tan acertado con los valores atípicos. Sin embargo podemos llegar a la conclusión que el algoritmo de estimación consigue predecir el resultado dentro de un rango bastante acertado.

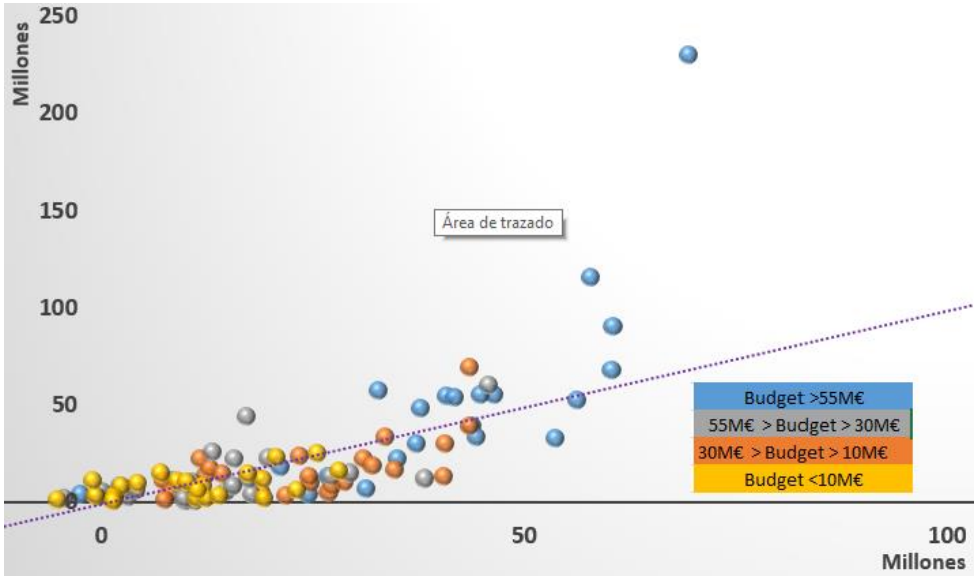
Imagen 27: Diferencia entre estimación y real de la variable dependiente por película



En segundo lugar, realizamos un análisis de las variables dependientes en función de tres variables independientes; dos de ellas se han comportado bien a la hora de realizar los algoritmos, y una que no ha entrado. Estas son: presupuesto, número de cines en los que se estrena, y posibilidad de ver la película en 3D o no.

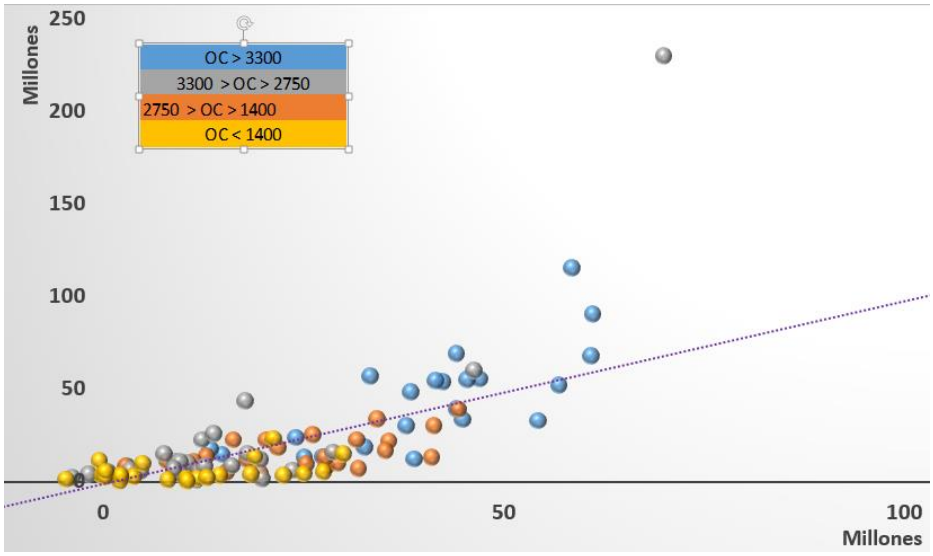
En lo que se refiere a *Budget*, se observa que las películas con un presupuesto más limitado son más predecibles; y sin embargo, aquellas de presupuesto alto presentan una alta variabilidad.

Imagen 28: Relación entre variable dependiente real y estimada por presupuesto (BUDG).



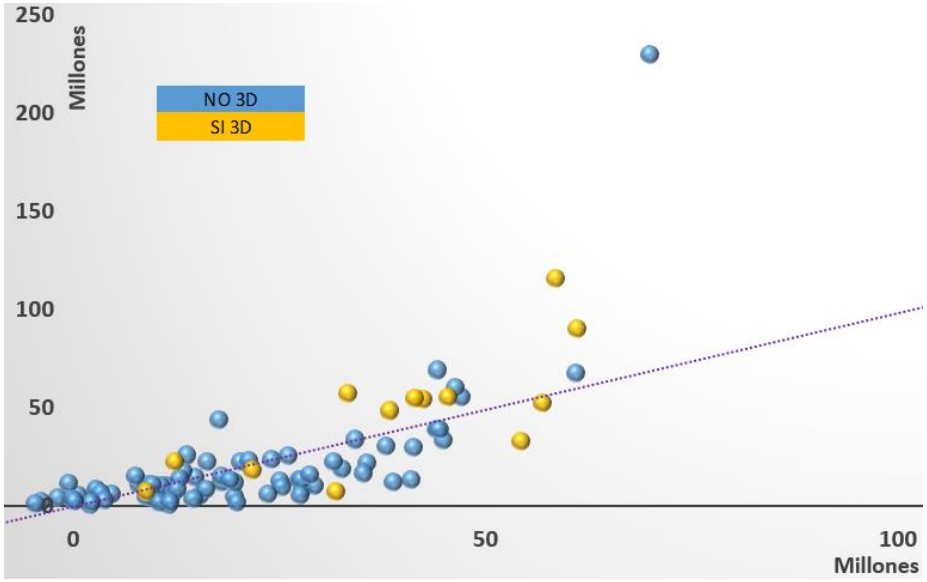
En lo que se refiere al número de cines en el que se estrena la película en su primer fin de semana, el resultado es muy parecido al anterior. No obstante en esta ocasión los valores con una cifra de cines baja (véase datos en amarillo en la imagen 29) tienden a quedarse minusvalorado, mientras que estima bien aquellos con valores promedios, y de nuevo la variabilidad es alta en los de mayor rango (véase datos en azul en la imagen 29).

Imagen 29: Relación entre variable dependiente real y estimada por número de cines (OC)



Para concluir, queríamos ver por qué una variable como 3D, que a priori podía pensarse que tendría importancia en la recaudación, no ha aparecido en ningún modelo. Tras representarla gráficamente (véase imagen 30) efectivamente se observa una variabilidad alta, tanto en los que sí tienen 3D como los que no lo tienen.

Imagen 30: Relación entre variable dependiente real y estimada por 3D



Como cierre, podemos apreciar como las redes sociales, a pesar de haber sido una fuente prometedora de información, no aportaron tanto al modelo como se esperaba, y consideramos que esto se debe a varios factores como:

- **Gestión insuficiente de RRSS:** muchas películas no tenían un canal oficial para su promoción, lo que nos llevó a extraer información del canal corporativo, y en este punto la información se mezclaba e impedía una lectura clara del flujo de información que realmente era generado solo por la película.
- **Volumen limitado de datos:** los datos que fueron recolectados no tenían el volumen esperado, y esto se debió a las limitaciones presentadas en su extracción. (Analisis del sector cinematográfico: Selección, extracción y análisis de variables provenientes de diversas fuentes web, como aporte inicial a la minería de datos, 2015, Fotunato Tarrona)
- **Mucho "ruido", pocas nueces:** la importancia de la información utilizada en el estudio, radica en la posibilidad de aporte que tenga una variable a determinar la posible intención del consumidor, dicho esto las redes sociales (al menos de la manera explotada) generan mucho "ruido", pero este ruido es difícilmente aprovechable al momento de predecir la intención de las personas. Para conseguir variables sólidas en este sentido, es necesario ser capaz de extraer y procesar un mayor número de tweets, para conseguir así
- **Dificultad para determinar qué es lo que realmente se habla de nuestro producto:** La dificultad de poder filtrar aquello que realmente nos interesa, hace que la información extraída se encuentre llena de incertidumbre, lo que reduce su valor al momento de generar conocimiento.

Dicho esto, no descartamos su valor, pero sí debemos plantear un reenfoque en la manera de cómo se realizó nuestra extracción.

9 RECOMENDACIONES PARA FUTURAS INVESTIGACIONES

Correspondería a un futuro investigador realizar un estudio más orientado al análisis de los resultados, desde el cual se puedan obtener conclusiones en cuanto a la importancia de cada variable que por limitaciones temporales y de espacio este trabajo no ha sido capaz de asumir.

Por otro parte, sería acertado incluir datos provenientes de otras redes sociales, como Facebook y Tumbler (esta última, se pudo apreciar que posee gran relevancia en el sector cinematográfico); así como reenfocar la extracción de información de twitter y YouTube, de tal manera que permita la reducción de la incertidumbre. Específicamente, recomendamos:

- **Twitter:**
    - Incrementar el número de *tweets* por película incluídos en el estudio. Los mensajes de tweeter muestras una variabilidad grande, y no todos muestran información relevante; es por ello que el volumen en este caso es importante y que la extracción, en el caso que sea necesario más de 7 días antes de la películas, debe ser planificada.
    - Afinar los procesos de selección para reducir el nivel de "ruido" no relacionado con el estudio.
- **YouTube:**
    - Incrementar el número de vídeos analizados, es decir, no tomar en cuenta solamente el vídeo más visto, sino al menos los 5 más vistos. Al realizar la extracción notamos cierta heterogeneidad, entre unas películas que solo se visualizaban en un único vídeo que recogía la mayoría de las visitas; y otras que repartían las visitas entre varios canales.

Además, sería interesante ampliar el número de observaciones, pues 400 es un número demasiado limitado como para que los modelos sean lo suficientemente robustos y precisos.

10 BIBLIOGRAFÍA

Boosting (machine learning), 2015. . Wikipedia Free Encycl.

Box Office Mojo [WWW Document], n.d. URL http://www.boxofficemojo.com/ (accessed 11.15.15).

Breiman, L., 2001. Random Forests. Mach. Learn. 45, 5–32. doi:10.1023/A:1010933404324

Clasificación por edades (cine), 2015. . Wikipedia Encicl. Libre.

Denil, M., Matheson, D., De Freitas, N., 2013. Narrowing the gap: Random forests in theory and in practice. 1310.1415.

DISTRIBUCIÓN F DE SNEDECOR [WWW Document], n.d. URL https://www.uv.es/ceaces/normaMu/f/f.htm (accessed 11.15.15).

Ensemble learning, 2015. . Wikipedia Free Encycl.

Google Study Can Predict Success Of Movies - Business Insider [WWW Document], n.d. URL http://www.businessinsider.com/google-study-can-predict-success-of-movies-2013-6 (accessed 11.15.15).

Heterocedasticidad, 2014. . Wikipedia Encicl. Libre.

IMDb - Movies, TV and Celebrities - IMDb [WWW Document], n.d. URL http://www.imdb.com/ (accessed 11.15.15).

Internet Archive: Wayback Machine [WWW Document], n.d. URL https://archive.org/web/ (accessed 11.15.15).

Javier Portela, 2007. Manual de Programación en SAS.

Leo Breiman, 1994. Bagging Predictors.

Michael Kearns, 1988. Thoughts on Hypothesis Boosting.

País, E.E., 2012. Twitter se convierte en EE UU en el termómetro económico del cine [WWW Document]. EL PAÍS. URL http://cultura.elpais.com/cultura/2012/05/03/actualidad/1336072604_870409.html (accessed 11.15.15).

Random forest - Wikipedia, la enciclopedia libre [WWW Document], n.d. URL https://es.wikipedia.org/wiki/Random_forest (accessed 11.16.15).

Regresión lineal - Wikipedia, la enciclopedia libre [WWW Document], n.d. URL https://es.wikipedia.org/wiki/Regresi%C3%B3n_lineal#Regresi.C3.B3n_lineal_m.C3.BAltiple (accessed 11.15.15).

SAS Institute, 2013. Getting started with SAS Enterprise Miner 13.1.

Schapire, R.E., 2003. The boosting approach to machine learning: An overview, in: Nonlinear Estimation and Classification. Springer, pp. 149–171.

Schapire, R.E., 1990. The strength of weak learnability. Mach. Learn. 5, 197–227.

Técnicas de regresión: Regresión Lineal Múltiple [WWW Document], n.d. URL https://www.fisterra.com/mbe/investiga/regre_lineal_multi/regre_lineal_multi.asp (accessed 11.15.15).

Tree (data structure), 2015. . Wikipedia Free Encycl.

Twitter [WWW Document], n.d. URL https://twitter.com/ (accessed 11.15.15).

Validación cruzada - Wikipedia, la enciclopedia libre [WWW Document], n.d. URL https://es.wikipedia.org/wiki/Validaci%C3%B3n_cruzada (accessed 11.15.15).

YouTube [WWW Document], n.d. URL https://www.youtube.com/ (accessed 11.15.15).

Google. "Quantifying Movie Magic with Google Search." *adwords.blogspot.com*, June 6, 2013). http://adwords.blogspot.com.es/2013/06/quantifying-movie-magic-with-google.html?m=1.

11 ANEXOS

11.1 TABLA DE ERRORES DE LOS MODELOS FINALES.

Hemos sombreado en rojo aquellos valores que han tenido un mayor error con cada semilla; en este sentido son los modelos Boosting los que nos aseguran mantener el error lo más bajo posible. Además, al igual que vemos en la gráfica de arriba, podemos apreciar.

Tabla 10: Errores de cada semilla para cada Modelo Óptimo por algoritmo.

ERROR PROMEDIO					
SEMILLA	RL	RN	RF	GBr	GBl
12345	3,08E+14	3,13E+14	3,13E+14	2,88E+14	2,95E+14
12346	3,44E+14	3,04E+14	3,40E+14	3,06E+14	2,97E+14
12347	2,99E+14	2,81E+14	3,02E+14	2,88E+14	2,96E+14
12348	3,29E+14	2,90E+14	3,18E+14	2,82E+14	2,83E+14
12349	3,58E+14	2,91E+14	3,41E+14	2,83E+14	2,95E+14
12350	2,99E+14	2,87E+14	3,12E+14	2,89E+14	2,95E+14
12351	3,17E+14	3,09E+14	3,12E+14	2,75E+14	2,72E+14
12352	3,09E+14	2,97E+14	3,18E+14	2,82E+14	2,90E+14
12353	3,17E+14	2,98E+14	3,35E+14	2,86E+14	2,91E+14
12354	3,25E+14	2,90E+14	3,28E+14	3,07E+14	2,96E+14
12355	3,07E+14	2,99E+14	3,01E+14	2,85E+14	2,89E+14
12356	3,20E+14	3,15E+14	3,11E+14	2,80E+14	2,85E+14
12357	3,61E+14	3,06E+14	3,49E+14	3,14E+14	3,18E+14
12358	3,32E+14	2,99E+14	3,20E+14	2,93E+14	2,99E+14
12359	3,24E+14	3,09E+14	3,02E+14	2,86E+14	2,93E+14
12360	3,19E+14	2,98E+14	3,10E+14	3,08E+14	3,07E+14
MAXIMO	**3,61E+14**	**3,15E+14**	**3,49E+14**	**3,14E+14**	**3,18E+14**
PROMEDIO	**3,23E+14**	**2,99E+14**	**3,19E+14**	**2,91E+14**	**2,94E+14**
MINIMO	**2,99E+14**	**2,81E+14**	**3,01E+14**	**2,75E+14**	**2,72E+14**